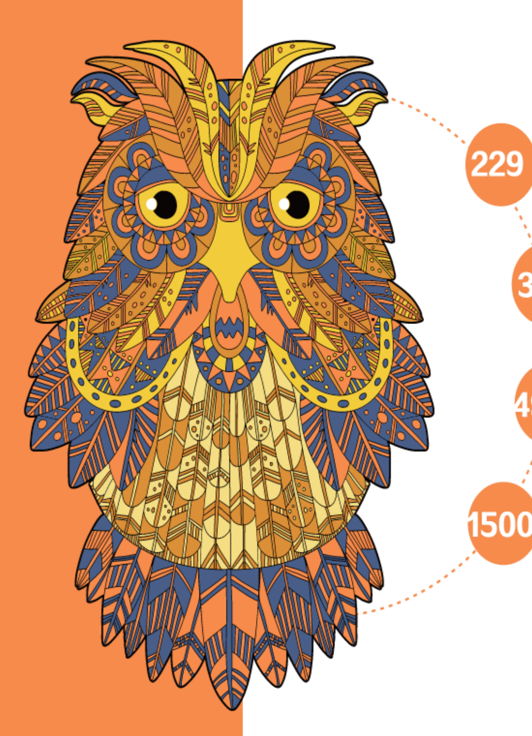
清华社"视频大讲堂"大系
网络开发视频大讲堂



229 节高清微视频＋269项在线微练习
移动端/PC端同步学习，QQ群/微信群随时答疑。

334 个实例案例分析＋220项实例源代码
速查、高效、实用，增强实战能力。

4900 个前端案例＋48本参考手册
先观摩，再临摹，高手案头常备，随时查阅提升。

1500 套网页模板＋12000个设计素材＋1036道前端面试真题
随用随取，提升设计效率，快速进阶开发高手行列。

CSS3+DIV 网页样式与布局从入门到精通

微课精编版

前端科技◎编著

线上资源，方便快捷

- 教学微视频（229节）
- 在线微练习（269项）
- 实例源代码（220项）

海量资源，可查可用

- 前端案例资源库（4900个）
- 面试题库（1036道）
- 网页模板库（1500套）
- 设计素材库（12000个）

清华大学出版社



CSS3+DIV 网页样式与布局从入门到精通

（微课精编版）

前端科技　编著

清华大学出版社

北　京

内容简介

《CSS3+DIV 网页样式与布局从入门到精通（微课精编版）》从初学者角度出发，通过通俗易懂的语言、丰富多彩的实例，详细介绍了 CSS3+DIV 前端开发技术及其应用。本书共 16 章，包括 CSS3 基础、使用 CSS3 选择器、使用 CSS 美化网页文本、使用 CSS 美化图像、使用 CSS 美化超链接、使用 CSS 美化列表、使用 CSS 美化表格、使用 CSS 美化表单、使用 DIV+CSS 设计网页、使用 HTML5+CSS3 设计网页、使用 CSS3 设计弹性布局、使用 CSS3 设计动态样式、使用 CSS3 设计响应式页面、使用 JavaScript 控制 CSS 样式、团购类型网站的布局与设计、掘客类型网站的布局与设计等内容。书中所有知识都结合具体实例进行介绍，代码注释详尽，读者可轻松掌握前端技术精髓，提升实际开发能力。

除纸质内容外，本书还配备了多样化、全方位的学习资源，主要内容如下。

- ☑ 229 节同步教学微视频
- ☑ 15000 项设计素材资源
- ☑ 220 项实例源代码
- ☑ 4800 个前端开发案例
- ☑ 334 个实例案例分析
- ☑ 48 本权威参考学习手册
- ☑ 269 个在线微练习
- ☑ 1036 道企业面试真题

本书可作为前端开发、移动开发入门者的自学用书，也可作为高等院校及相关培训机构的教学参考用书。

图书在版编目（CIP）数据

CSS3+DIV 网页样式与布局从入门到精通：微课精编版 / 前端科技编著. —北京：清华大学出版社，2018
（清华社“视频大讲堂”大系 网络开发视频大讲堂）
ISBN 978-7-302-50199-2

Ⅰ. ①C…　Ⅱ. ①前…　Ⅲ. ①网页制作工具　Ⅳ. ①TP393.092.2

中国版本图书馆 CIP 数据核字（2018）第 114530 号

责任编辑： 贾小红
封面设计： 李志伟
版式设计： 楠竹文化
责任校对： 毛姗姗
责任印制：

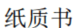
出版发行： 清华大学出版社
网　　址： http://www.tup.com.cn，http://www.wqbook.com
地　　址： 北京清华大学学研大厦 A 座　　**邮　　编：** 100084
社 总 机： 010-62770175　　**邮　　购：** 010-62786544
投稿与读者服务： 010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈： 010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销： 全国新华书店
开　　本： 203mm×260mm　　**印　　张：** 34.75　　**字　　数：** 905 千字
版　　次： 2018 年 12 月第 1 版　　**印　　次：** 2018 年 12 月第 1 次印刷
定　　价： 89. 80 元

产品编号：078929-01

如何使用本书 

本书提供了多样化、全方位丰富的学习资源，帮助读者轻松掌握 CSS3+DIV 技术，从小白快速成长为前端开发高手。





纸质书



视频讲解



在线练习



电子书

手机端+PC 端，线上线下同步学习

1. 获取学习权限

学习本书前，请先刮开图书封底的二维码涂层，使用手机扫描，即可获取本书资源的学习权限。再扫描正文章节对应的二维码，就可以观看视频讲解和在线练习提升，全程易懂、好学、速查、高效、实用。



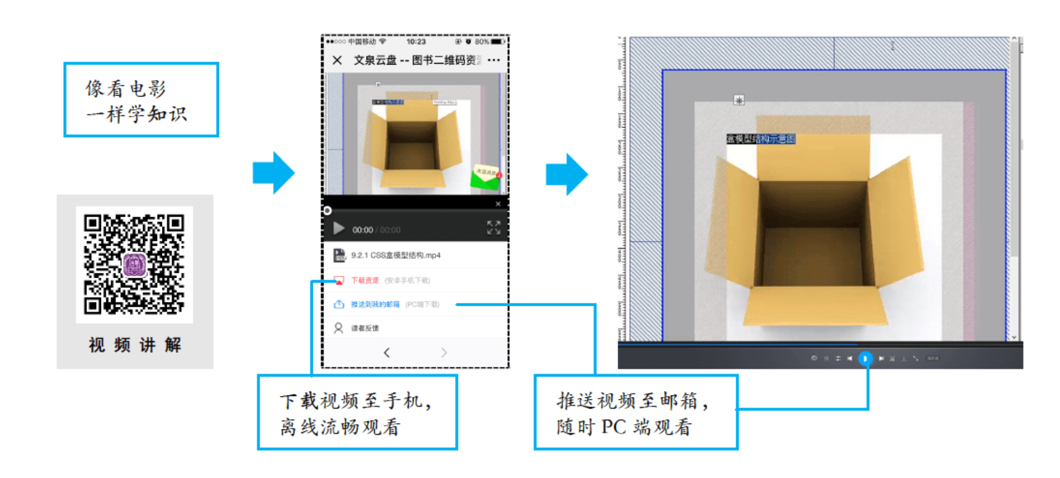
2. 观看视频讲解

对于初学者来说，精彩的知识讲解和透彻的实例解析能够引导其快速入门，轻松理解和掌握知识要点。本书中几乎所有案例都录制了视频，可以使用手机在线观看，也可以离线观看，还可以推送到计算机上大屏幕观看。



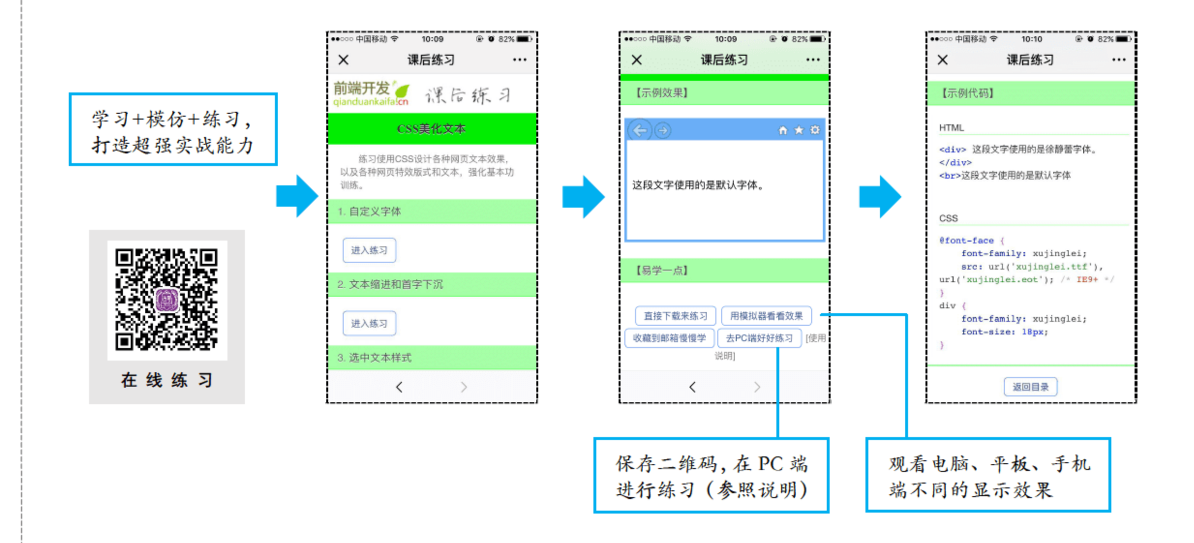






3. 进行线上练习

为方便读者巩固基础知识，提升实战能力，本书附赠了大量的前端练习题目。读者扫描各章最后的“在线练习”二维码，即可通过反复的实操训练加深对知识的领悟程度。



4. 其他 PC 端资源下载方式

除了前面介绍过的可以直接将视频、在线练习等资源推送到邮箱之外，还提供了如下几种 PC 端资源获取方式。

☑ 登录清华大学出版社官方网站（www.tup.com.cn），在对应图书页面下查找资源的下载方式。

☑ 申请加入 QQ 群、微信群，获得资源的下载方式。

☑ 扫描图书封底“文泉云盘”二维码，获得资源的下载方式。



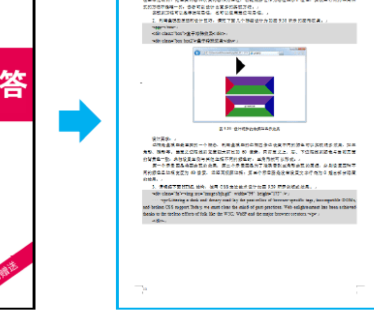


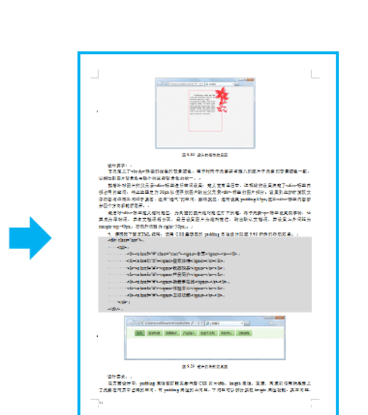
小白实战电子书

为方便读者全面提升，本书赠送了“前端开发百问百答”小白学习电子书。这些内容精挑细选，希望成为您学习路上的好帮手，关键时刻解您所需。

扫描图书封底的二维码，可在手机、平板上学习小白手册内容。







从小白到高手的蜕变

谷歌的创始人拉里·佩奇说过，如果你刻意练习某件事超过 10000 个小时，那么你就可以达到世界级。

因此，不管您现在是怎样的前端开发小白，只要您按着下面的步骤来学习，假以时日，您会成为令自己惊讶的技术大咖。

（1）扎实的基础知识+大量的中小实例训练+有针对性地做一些综合案例。

（2）大量的项目案例观摩、学习、操练，塑造一定的项目思维。

（3）善于借用他山之石，对一些成熟的开源代码、设计素材拿来就用，学会站在巨人的肩膀上。

（4）有工夫多参阅一些官方权威指南，拓展自己对技术的理解和应用能力。

（5）最为重要的是，多与同行交流，在切磋中不断进步。

书本厚度有限，学习空间无限。纸张价格有限，知识价值无限。希望本书能帮您真正收获学习的乐趣和知识。最后，祝您阅读快乐！

前　　言



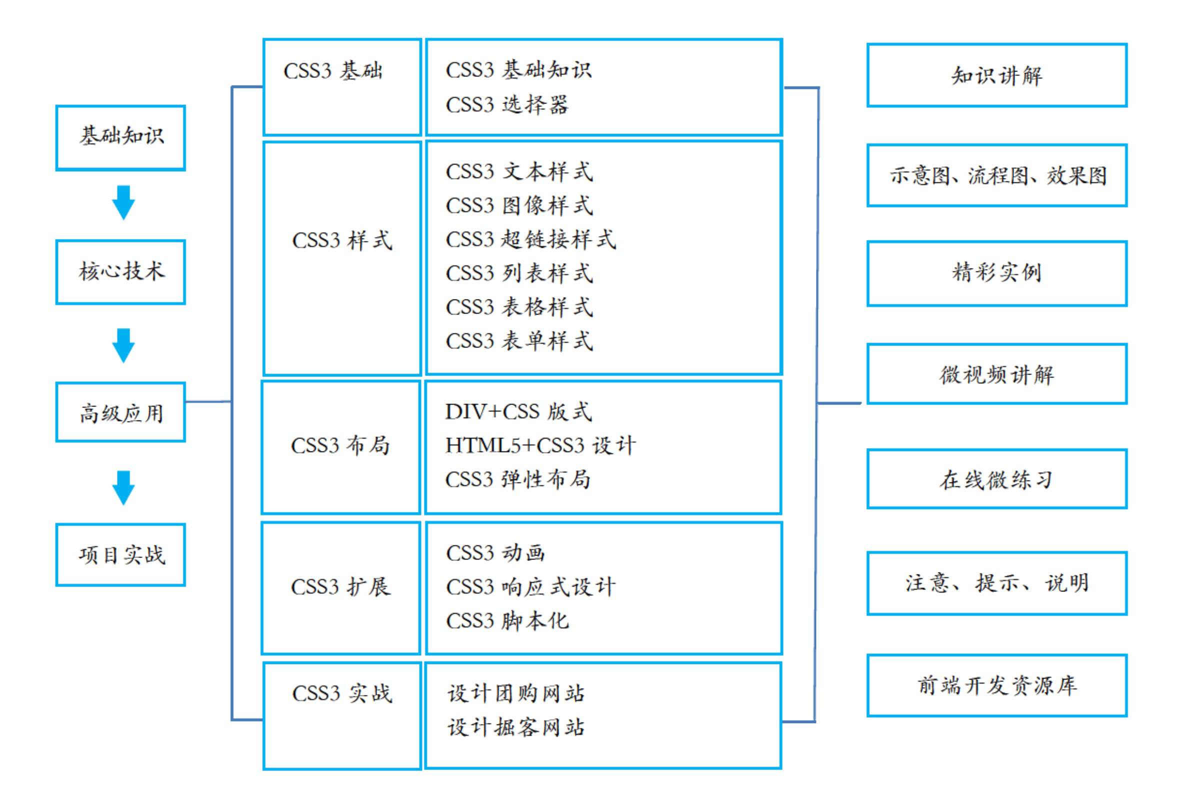
Preface

“网络开发视频大讲堂”系列丛书于2013年5月出版，因其编写细腻、讲解透彻、实用易学、配备全程视频等，备受读者欢迎。丛书累计销售近20万册，其中，《HTML5+CSS3从入门到精通》累计销售10万册。同时，系列书被上百所高校选为教学参考用书。

本次改版，在继承前版优点的基础上，进一步对图书内容进行了优化，选择面试、就业最急需的内容，重新录制了视频，同时增加了许多当前流行的前端技术，提供了“入门学习→实例应用→项目开发→能力测试→面试”等各个阶段的海量开发资源库，实战容量更大，以帮助读者快速掌握前端开发所需要的核心精髓内容。

网页技术层出不穷，日新月异，但是不管采用什么技术，用户通过浏览器看到的网页都是由HTML和CSS构成的。因此，HTML和CSS技术是网页制作技术的基础和核心。本书将重点讲解如何使用CSS设计网页样式和版式。

本书内容







本书特点

1. 由浅入深，编排合理，实用易学

本书面向零基础的初学者，通过“一个知识点+一个例子+一个结果+一段评析+一个综合应用”的写作模式，全面、细致地讲述了CSS3+DIV实际开发中所需的各类知识，由浅入深，循序渐进。同时，本书展示了许多Web时代备受欢迎的新知识，读者可学习到与HTML5相关的一些非常实用、流行的技术。

2. 跟着案例和视频学，入门更容易

跟着例子学习，通过训练提升，是初学者最好的学习方式。本书案例丰富详尽，多达334个，且都附有详尽的代码注释及清晰的视频讲解。跟着这些案例边做边学，可以避免学到的知识流于表面、限于理论，尽情感受编程带来的快乐和成就感。

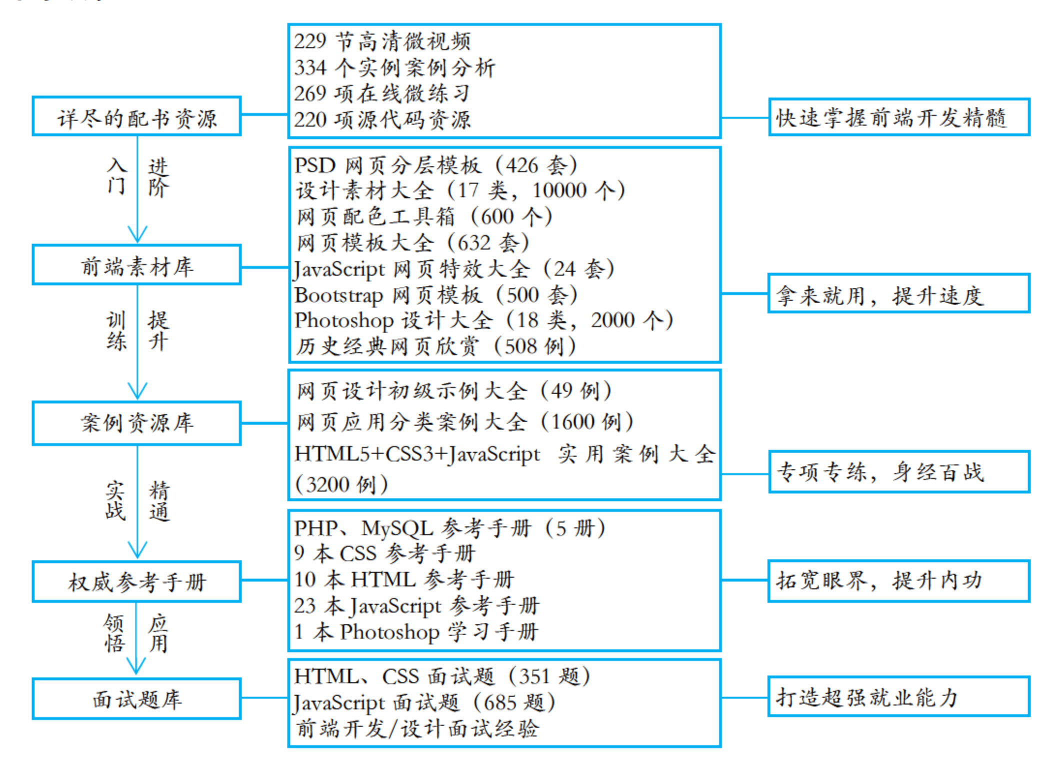
3. 丰富线上资源，多元化学习体验

为了传递更多知识，本书力求突破传统纸质书的厚度限制。本书提供了丰富的线上微资源，通过手机扫码，读者可随时观看讲解视频，在线练习强化提升，全程便捷、高效，感受不一样的学习体验。

4. 精彩栏目，易错点、重点、难点贴心提醒

本书根据初学者特点，在一些易错点、重点、难点位置精心设置了“注意”“提示”等小栏目。通过这些小栏目，读者会更留心相关的知识点和概念，绕过陷阱，掌握很多应用技巧。

本书资源







读者对象

☑ 零基础的编程自学者。
☑ 相关培训机构的老师和学生。
☑ 大中专院校的老师和学生。
☑ 参加毕业设计的学生。
☑ 初、中级程序开发人员。

读前须知

作为入门书籍，本书知识点比较庞杂，所以不可能面面俱到。技术学习的关键是方法，本书在很多实例中体现了方法的重要性，读者只要掌握了各种技术的运用方法，在学习更深入的知识中可大大提高自学的效率。

本书提供了大量示例，需要用到 IE、Firefox、Chrome、Opera 等主流浏览器的测试和预览。因此，为了测试示例或代码，读者需要安装上述类型的最新版本浏览器，各种浏览器在 CSS3 的表现上可能会稍有差异。

限于篇幅，本书示例没有提供完整的 HTML 代码，读者应该补充完整的 HTML 结构，然后进行测试练习，或者直接参考本书提供的下载源代码，边学边练。

为了给读者提供更多的学习资源，本书提供了很多参考链接，许多本书无法详细介绍的问题都可以通过这些链接找到答案。由于这些链接地址会因时间而有所变动或调整，所以在此说明，这些链接地址仅供参考，本书无法保证所有的这些地址是长期有效的。

读者服务

学习本书时，请先扫描封底的权限二维码（需要刮开涂层）获取学习权限，然后即可免费学习书中的所有线上线下资源。

本书所附赠的超值资源库内容，读者可登录清华大学出版社网站（www.tup.com.cn），在对应图书页面下获取其下载方式。也可扫描图书封底的“文泉云盘”二维码，获取其下载方式。

本书提供 QQ 群（668118468、697651657）、微信公众号（qianduankaifa_cn）、服务网站（www.qianduankaifa.cn）等互动渠道，提供在线技术交流、学习答疑、技术资讯、视频课堂、在线勘误等功能。在这里，您可以结识大量志同道合的朋友，在交流和切磋中不断成长。

读者对本书有什么好的意见和建议，也可以通过邮箱（qianduanjiaoshi@163.com）发邮件给我们。





Note

关于作者

前端科技是由一群热爱 Web 开发的青年骨干教师和一线资深开发人员组成的一个团队，主要从事 Web 开发、教学和培训。参与本书编写的人员包括咸建勋、奚晶、文菁、李静、钟世礼、袁江、甘桂萍、刘燕、杨凡、朱砚、余乐、邹仲、余洪平、谭贞军、谢党华、何子夜、赵美青、牛金鑫、孙玉静、左超红、蒋学军、邓才兵、陈文广、李东博、林友赛、苏震巍、崔鹏飞、李斌、郑伟、邓艳超、胡晓霞、朱印宏、刘望、杨艳、顾克明、郭靖、朱育贵、刘金、吴云、赵德志、张卫其、李德光、刘坤、彭方强、雷海兰、王鑫铭、马林、班琦、蔡霞英、曾德剑等。

尽管已竭尽全力，但由于水平有限，书中疏漏和不足之处在所难免，欢迎各位读者朋友批评、指正。

编者

2018 年 8 月

目　　录



Contents










Not







Note






Not




Note



CSS3 基础

CSS3 是 CSS 规范的最新版本，在 CSS2 基础上增加了很多新功能，以帮助开发人员解决一些实际问题，如圆角、多背景、透明度、阴影等功能。本章将简单介绍 CSS 的基础知识，读者可以初步了解 CSS 的基本用法。

【学习要点】

⏭ 了解 CSS 的发展历史。

⏭ 正确使用 CSS。

⏭ 了解 CSS 的基本特性。





1.1　CSS 发展历史

CSS 是 Cascading Style Sheet 的缩写，中文翻译为层叠样式表。CSS 控制 HTML 标签的显示样式，负责设计页面效果。使用 CSS 能够实现页面的内容与表现分离，提高工作效率。

早期的 HTML 标签含有大量的显示属性，这带来了一个问题：网页结构和显示样式混在一起，使网页代码混乱不堪，代码冗余增加了带宽负担，代码维护也变得苦不堪言。

1994 年，哈坤首次提出了 CSS 的想法，1995 年他与波斯一起展示这个建议。

1996 年，W3C 正式推出 CSS 语言，并于同年 12 月发布 CSS 1.0 版本（http://www.w3.org/TR/CSS1/）。

1998 年 5 月，CSS 2.0 版本正式发布（http://www.w3.org/TR/CSS2/）。

2001 年 5 月 23 日，W3C 完成 CSS3 版本的工作草案，在该草案中制订了 CSS3 发展路线图，路线图详细列出了所有模块，并计划在未来逐步进行规范。

2002 年，由于各方对 CSS3 标准分歧太大，W3C 开始开发 CSS 2.1 版本。CSS 2.1 是 CSS 2.0 的修订版，它纠正了 CSS 2.0 版本中的一些错误，并且更精确地描述了 CSS 的浏览器实现。

2004 年，CSS 2.1 正式发布，2006 年年底得到完善。CSS 2.1 是目前最流行、获得浏览器支持最完整的版本，它更准确地反映了 CSS 当前的状态。

到目前为止，CSS3 还没有推出正式的完整版，但是已经陆续推出了不同的模块，这些模块已经被大部分浏览器支持或部分实现。

CSS3 属性支持情况如图 1.1 所示（http://fmbip.com/litmus/）。可以看出，完全支持 CSS3 属性的浏览器为 Chrome 和 Safari。

平台	MAC				WIN								
浏览器	CHROME	FIREFOX	OPERA	SAFARI	CHROME	FIREFOX		OPERA		SAFARI	IE		
版本	5	3.6	10.1	4	4	3.6	3	10	10.5	4	6	7	8
RGBA	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✘	✘	✘
HSLA	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✘	✘	✘
Multiple Backgrounds	✔	✔	✘	✔	✔	✔	✘	✘	✔	✔	✘	✘	✘
Border Image	✔	✔	✘	✔	✔	✔	✘	✘	✔	✔	✘	✘	✘
Border Radius	✔	✔	✘	✔	✔	✔	✔	✘	✔	✔	✘	✘	✘
Box Shadow	✔	✔	✘	✔	✔	✔	✘	✘	✔	✔	✘	✘	✘
Opacity	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✘	✘	✘
CSS Animations	✔	✘	✘	✔	✔	✘	✘	✘	✘	✔	✘	✘	✘
CSS Columns	✔	✔	✘	✔	✔	✔	✔	✘	✘	✔	✘	✘	✘
CSS Gradients	✔	✔	✘	✔	✔	✔	✘	✘	✘	✔	✘	✘	✘
CSS Reflections	✔	✘	✘	✔	✔	✘	✘	✘	✘	✔	✘	✘	✘
CSS Transforms	✔	✔	✘	✔	✔	✔	✘	✘	✔	✔	✘	✘	✘
CSS Transforms 3D	✔	✘	✘	✔	✔	✘	✘	✘	✘	✔	✘	✘	✘
CSS Transitions	✔	✘	✘	✔	✔	✘	✘	✘	✔	✔	✘	✘	✘
CSS FontFace	✔	✔	✔	✔	✔	✔	✘	✔	✔	✔	✔	✔	✔

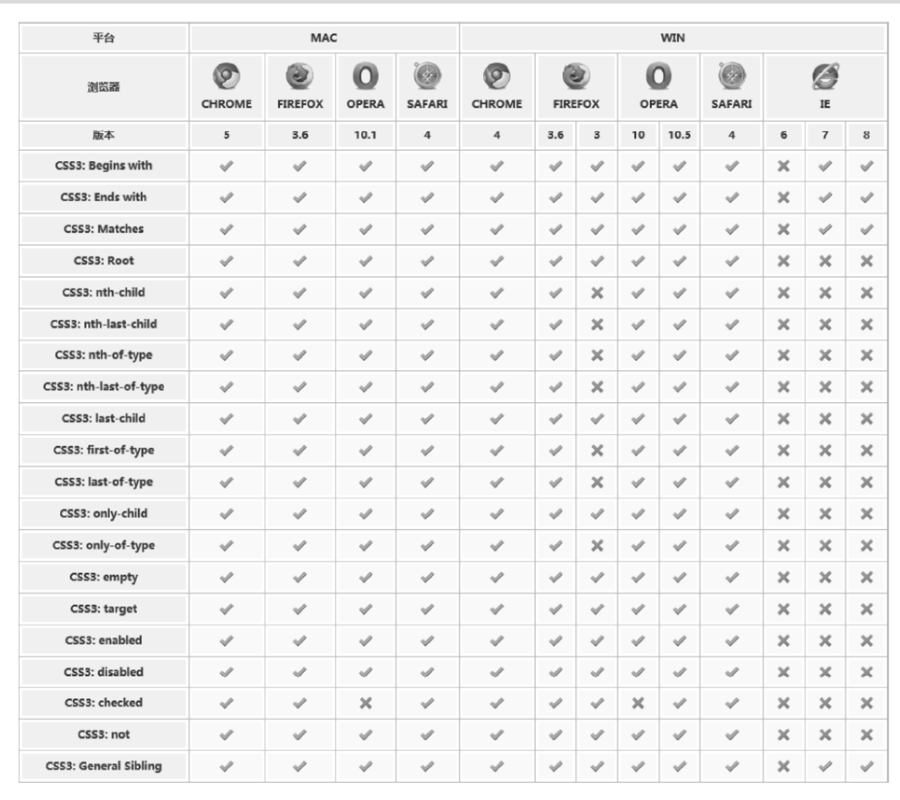
图 1.1　CSS3 属性支持列表





CSS3 选择器支持情况如图 1.2 所示（http://fmbip.com/litmus/）。除了 IE 系列版本和 Firefox 3，其他浏览器几乎全部支持，如 Chrome、Safari、Firefox 和 Opera。

提示：部分浏览器允许使用私有属性支持 CSS3 的新特性。例如，Webkit 类型浏览器（如 Safari、Chrome）的私有属性是以-webkit-前缀开始，Gecko 类型浏览器（如 Firefox）的私有属性是以-moz-前缀开始，Konqueror 类型浏览器的私有属性是以-khtml-前缀开始，Opera 浏览器的私有属性是以-o-前缀开始，而 Internet Explorer 浏览器的私有属性是以-ms-前缀开始（目前只有 IE8+支持-ms-前缀）。



平台	MAC				WIN								
浏览器	CHROME	FIREFOX	OPERA	SAFARI	CHROME	FIREFOX		OPERA		SAFARI	IE		
版本	5	3.6	10.1	4	4	3.6	3	10	10.5	4	6	7	8
CSS3: Begins with	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✘	✔	✔
CSS3: Ends with	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✘	✔	✔
CSS3: Matches	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✘	✔	✔
CSS3: Root	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✘	✘	✘
CSS3: nth-child	✔	✔	✔	✔	✔	✔	✘	✔	✔	✔	✘	✘	✘
CSS3: nth-last-child	✔	✔	✔	✔	✔	✔	✘	✔	✔	✔	✘	✘	✘
CSS3: nth-of-type	✔	✔	✔	✔	✔	✔	✘	✔	✔	✔	✘	✘	✘
CSS3: nth-last-of-type	✔	✔	✔	✔	✔	✔	✘	✔	✔	✔	✘	✘	✘
CSS3: last-child	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✘	✘	✘
CSS3: first-of-type	✔	✔	✔	✔	✔	✔	✘	✔	✔	✔	✘	✘	✘
CSS3: last-of-type	✔	✔	✔	✔	✔	✔	✘	✔	✔	✔	✘	✘	✘
CSS3: only-child	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✘	✘	✘
CSS3: only-of-type	✔	✔	✔	✔	✔	✔	✘	✔	✔	✔	✘	✘	✘
CSS3: empty	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✘	✘	✘
CSS3: target	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✘	✘	✘
CSS3: enabled	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✘	✘	✘
CSS3: disabled	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✘	✘	✘
CSS3: checked	✔	✔	✘	✔	✔	✔	✔	✘	✔	✔	✘	✘	✘
CSS3: not	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✘	✘	✘
CSS3: General Sibling	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✘	✔	✔

图 1.2　CSS3 选择器支持列表

1.2　使用 CSS

CSS 代码可以在任何文本编辑器中打开和编辑，下面介绍 CSS 的基本语法和用法。

1.2.1　CSS 样式



样式是 CSS 的最小语法单元，每个样式包含两部分内容：选择器和声明，如图 1.3 所示。




Note

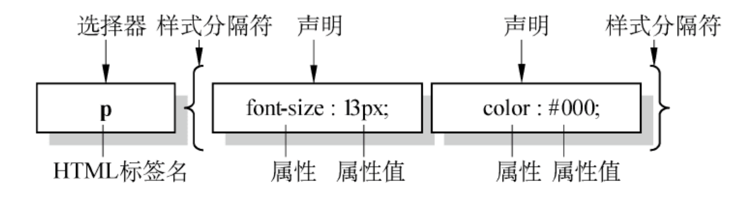


图 1.3　CSS 样式的基本结构

☑ 选择器：选择器告诉浏览器该样式将作用于页面中哪些对象，这些对象可以是一个或多个标签，也可以是某个具体对象等。

☑ 声明：声明命令浏览器如何渲染指定的对象。声明由两部分组成，即属性和属性值，并用分号标识一个声明的结束，样式中最后一个声明可以省略分号。样式中可以包含一个或多个声明，所有声明被放置在一对大括号内，紧邻选择器后面。

- 属性：属性是 CSS 预定义的样式选项。属性名由一个单词或多个单词组成，多个单词之间通过连字符相连。这样能够很直观地表示属性所要设置的样式效果。
- 属性值：定义属性显示效果的参数，包括数值和单位，或者关键字。

【示例 1】定义网页字体大小为 12 像素，字体颜色为深灰色，则设置如下样式。

```
body{font-size: 12px; color: #CCCCCC;}
```

多个样式可以并列在一起，不需要考虑如何进行分隔。

【示例 2】定义段落文本的背景色为紫色，则可以在上面样式基础上定义如下样式。

```
body{font-size: 12px; color: #CCCCCC;}
p{background-color: #FF00FF;}
```

提示：由于 CSS 语言忽略空格（除了选择器内部的空格外），因此可以利用空格来格式化 CSS 源代码，则上面代码可以进行如下美化。

```
body {
    font-size: 12px;
    color: #CCCCCC;
}
p { background-color: #FF00FF; }
```

这样在阅读 CSS 源代码时就一目了然了，既方便阅读，也更容易维护。


视 频 讲 解

1.2.2　应用 CSS 样式

应用 CSS 样式的方法包括 3 种：行内样式、内部样式和外部样式，下面分别进行说明。

1. 行内样式

行内样式就是把 CSS 样式直接放在代码行内的标签中，一般都是放入标签的 style 属性中，由于





行内样式直接插入标签中，故是最直接的一种方式。

【示例 1】在本示例中，针对段落、<h2>标签、<em>标签、<strong>标签以及<div>标签分别应用了 CSS 行内样式，代码如下，页面演示效果如图 1.4 所示。

```
<p style="background-color:#999900">行内元素，控制段落-1</p>
<h2 style="background-color:#FF6633">行内元素，h2 标题元素</h2>
<p style="background-color:#999900">行内元素，控制段落-2</p>
<strong style="font-size:30px;">行内元素，strong 比 em 效果要强</strong>
<div style="background-color:#66CC99; color:#993300;height:30px; line-height: 30px;">行内元素，div 块级元素</div>
<em style="font-size:2em;">行内元素，em 强调</em>
```

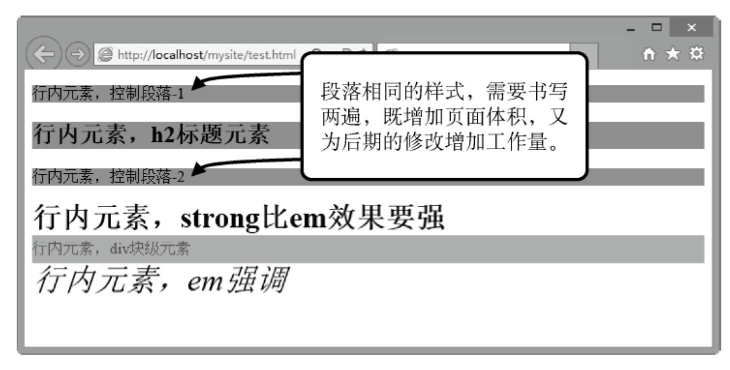


图 1.4　行内样式的应用

2. 内部样式

内部样式是将 CSS 代码写在 HTML 的<style>标签中，其特点是该样式只能在此页有效。

【示例 2】在本示例中，通过内部样式为段落设置显示样式。页面演示效果如图 1.5 所示。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<style type="text/css">
p {
    text-align: left;                /* 文本左对齐 */
    font-size: 14px;                 /* 字体大小 14 像素 */
    line-height: 25px;               /* 行高 25 像素 */
    text-indent: 2em;                /* 首行缩进 2 个文字大小空间 */
    width: 500px;                    /* 段落宽度 500 像素 */
    margin: 0 auto;                  /* 浏览器居中 */
    margin-bottom: 20px;             /* 段落下边距 20 像素 */
```






```
    }
</style>
</head>
<body>
<p>“百度”这一公司名称便来自宋词“众里寻他千百度”。百度公司会议室名为青玉案，即是这首词的词牌。而“熊掌”图标的想法来源于“猎人巡迹熊爪”的刺激，与李博士的“分析搜索技术”非常相似，从而构成百度的搜索概念，也最终成为百度的图标形象。在这之后，由于在搜索引擎中，大都有动物形象来标志，如 SOHU 的狐，如 GOOGLE 的狗，而百度也便顺理成章称作了熊。百度熊也便成了百度公司的形象物。</p>
<p>在百度那次更换 LOGO 的计划中，百度给出的 3 个新 LOGO 设计方案在网民的投票下，全部被否决，更多的网民将选票投给了原有的熊掌标志。</p>
<p>此次更换LOGO的行动共进行了3轮投票，直到第2轮投票结束，新的笑脸LOGO都占据了绝对优势。但到最后一轮投票时，原有的熊掌标志却戏剧性地获得了最多的网民选票，从而把 3 个新 LOGO 方案彻底否决。</p>
</body>
</html>
```

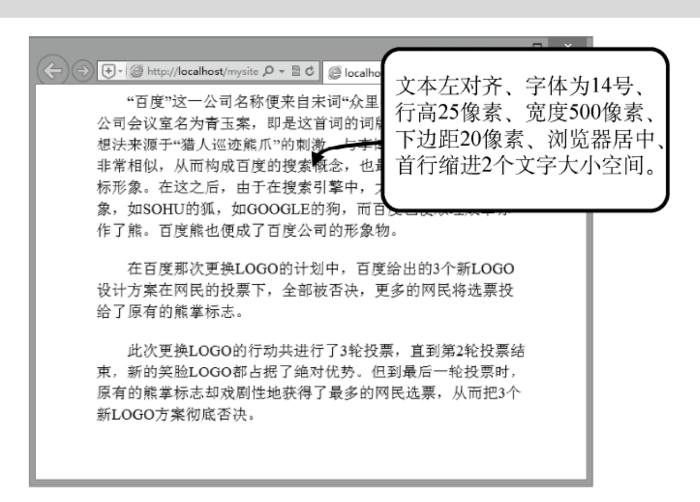


图 1.5　内部样式的应用

注意：<style>标签不仅可以定义 CSS 样式，还可以定义 JavaScript 脚本。当<style>标签的 type 属性值为 text/css 时，内部代码为 CSS 样式；当<style>标签的 type 属性值为 text/javascript 时，内部代码为 JavaScript 脚本。

3. 外部样式

外部样式通过 HTML 的<link>标签，将外部样式表文件链接到 HTML 文档中，这是最实用的方式。这种方法将 HTML 文档和 CSS 文件完全分离，实现结构层和表示层的彻底分离，增强网页结构的扩展性和 CSS 样式的可维护性。





【示例 3】在本示例中，使用链接式为 HTML 代码应用样式，书写、更改方便。页面演示效果如图 1.6 所示。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<link href="lianjie.css" type="text/css" rel="stylesheet" />
<link href="lianjie-2.css" type="text/css" rel="stylesheet" />
</head>
<body>
<p>我是被 lianjie-2.css 文件控制的，楼下的你呢？？</p>
<h3>楼上的,<span>lianjie.css</span>文件给我穿的花衣服。</h3>
</body>
</html>
```



图 1.6　外部样式的应用

在上面的示例中，通过 link 链接两个外部 CSS 文件，其中将公共样式放入一个 CSS 文件，当前页面样式放入另一个 CSS 文件。

☑　lianjie.css 文件代码如下。

```
h3{
    font-weight:normal;                 /* 取消标题默认加粗效果 */
    background-color:#66CC99;           /* 设置背景色 */
    height:50px;                        /* 设置标签的高度 */
    line-height:50px;                   /* 设置标签的行高*/
    }
```





```
span{
    color:#FF0000;                          /* 设置字体颜色 */
    font-weight:bold;                       /* 字体加粗 */
    }
```

☑ lianjie-2.css 文件代码如下。

```
p{
    color:#FF3333;                          /* 设置字体颜色 */
    font-weight:bold;                       /* 字体加粗 */
    border-bottom:3px dashed #009933;       /* 设置下边框线 */
    line-height:30px;                       /* 设置行高 */
    }
```

CSS 文件可以使网站所有页面样式统一，便于管理，减少代码和维护时间。当修改 CSS 文件时，所有应用此 CSS 文件的页面样式都将更新。



视频讲解

1.2.3 CSS 样式表

一个或多个 CSS 样式可以组成一个样式表。样式表包括内部样式表和外部样式表，它们没有本质区别，都是由一个或者多个样式组成。

1. 内部样式表

内部样式表包含在<style>标签内，一个<style>标签就表示一个内部样式表。而通过标签的 style 属性定义的样式属性就不是样式表。如果一个网页文档中包含多个<style>标签，就表示该文档包含多个内部样式表。

2. 外部样式表

如果 CSS 样式被放置在网页文档外部的文件中，则称为外部样式表，一个 CSS 样式表文档就表示一个外部样式表。实际上，外部样式表也就是文本文件，扩展名为.css。当把 CSS 样式代码复制到一个文本文件中后，另存为.css 文件，则它就是一个外部样式表。如图 1.7 所示就是禅意花园的外部样式表（http://www.csszengarden.com/）。

可以在外部样式表文件顶部定义 CSS 源代码的字符编码。

【示例】下面的代码定义样式表文件的字符编码为中文简体。

```
@charset "gb2312";
```

如果不设置 CSS 文件的字符编码，可以保留默认设置，则浏览器会根据 HTML 文件的字符编码来解析 CSS 代码。








```
/* css Zen Garden default style v1.02 */
/* css released under Creative Commons License - http://creativecommons.org/licenses/by-nc-sa/1.0/  */

/* This file based on 'Tranquille' by Dave Shea */
/* You may use this file as a foundation for any new work, but you may find it easier to start from scratch.
*/
/* Not all elements are defined in this file, so you'll most likely want to refer to the xhtml as well. */

/* Your images should be linked as if the CSS file sits in the same folder as the images. ie. no paths. */


/* basic elements */
html {
        margin: 0;
        padding: 0;
        }
body {
        font: 75% georgia, sans-serif;
        line-height: 1.88889;
        color: #555753;
        background: #fff url(http://csszengarden.com/001/blossoms.jpg) no-repeat bottom right;
        margin: 0;
        padding: 0;
        }
p {
        margin-top: 0;
        text-align: justify;
        }
```

图 1.7　禅意花园外部样式表文件

1.2.4　导入外部样式表

外部样式表必须导入网页文档中，才能够被浏览器识别和解析。外部样式表文件可以通过以下两种方法导入 HTML 文档中。

1. 使用<link>标签导入

使用<link>标签导入外部样式表文件，代码如下。

```
<link href="001.css" rel="stylesheet" type="text/css" />
```

其中 href 属性设置外部样式表文件的地址，可以是相对地址，也可以是绝对地址。rel 属性定义该标签关联的是样式表标签。type 属性定义文档的类型，即为 CSS 文本文件。

一般在定义<link>标签时，应定义 3 个基本属性，其中 href 是必须设置属性。具体说明如下。

☑　href：定义样式表文件 URL。

☑　type：定义导入文件类型，同 style 元素一样。

☑　rel：用于定义文档关联，这里表示关联样式表。

也可以在 link 元素中添加 title 属性，设置可选样式表的标题，即当一个网页文档导入了多个样式表后，可以通过 title 属性值选择所要应用的样式表文件。

另外，title 属性与 rel 属性存在联系，按 W3C 组织的计划，未来的网页文档会使用多个 link 元素导入不同的外部文件，如样式表文件、脚本文件、主题文件，甚至可以包括个人自定义的其他补充文件。导入这么多不同类型、名称各异的文件后，可以使用 title 属性进行选择，这时 rel 属性的作用就显现出来了，它可以指定网页文件初始显示时应用的导入文件类型，目前只能关联 CSS 样式表类型。

外部样式是 CSS 应用最佳方案，一个样式表文件可以被多个网页文件引用，同时一个网页文件可以导入多个样式表，方法是重复使用 link 元素导入不同的样式表文件。

2. 使用@import 关键字导入

在<style>标签内使用@import 关键字导入外部样式表文件，代码如下。

```
<style type="text/css">
```



```
@import url("001.css");
</style>
```



Note

在@import关键字后面，利用url()函数包含具体的外部样式表文件的地址。

提示：两种导入样式表的方法比较如下。

- link属于HTML标签，而@import是CSS提供的。
- 加载页面时，link会同时被加载，而@import引用的CSS会等到页面加载完再加载。
- @import只在IE 5以上才能识别，而link是HTML标签，无兼容问题。
- link方式的样式的权重高于@import的权重。

因此，一般推荐link导入样式表的方法，@import可以作为补充方法使用。

1.2.5 CSS注释

视频讲解

在CSS中增加注释很简单，所有被放在“/*”和“*/”分隔符之间的文本信息都被称为注释。

【示例1】整段代码单行注释。

```
/* 下面这段代码的作用是建立网页布局 start */
.head{width:960px;}
/* 下面这段代码的作用是建立网页布局 end */
```

【示例2】整段代码多行注释。

```
/* 下面这段代码的作用是建立网页布局
它包括头部和尾部宽度设置 start */
.head{width:960px;}
.footer{width:960px;}
/* 下面这段代码的作用是建立网页布局
它包括头部和尾部宽度设置 end */
```

【示例3】单行代码注释。

```
.p {
        color: #ff7000;                    /* 字体颜色设置 */
        height：30px;                      /* 段落高度设置 */
}
```

上面给出了整段代码单行注释、整段代码多行注释以及单行代码注释，它们的共同点同时也是CSS代码注释的要求：注释语句以“/*”开始，以“*/”结束，中间加入注释内容。下面的例子是进行CSS代码应用并查看浏览器下的效果。

【示例4】在本示例中，将段落和标题分别添加CSS注释。页面演示效果如图1.8所示。

```
<style type="text/css">
/* 关于段落的注释  开始*/
```



```
.STYLE1 {
        color: #009900;                  /* 字体颜色是绿色的 */
}
.STYLE2 {
        font-size: 18px;                 /* 字体大小为 18 像素 */
        color: #FF3300;                  /* 字体颜色是红色的*/
        font-weight: bold;               /* 字体进行了加粗 */
}
/* 关于段落的注释　结束*/
/* 标签设置注释　开始*/
.STYLE3 {
        color: #0000FF;                  /* 字体颜色为蓝色 */
        font-family: "黑体";             /* 字体为黑体 */
        font-style: italic;              /* 字体效果为倾斜 */
}
/* 标签设置注释　结束*/
</style>

<p class="STYLE1">段落设置一</p>
<p class="STYLE2">段落设置二</p>
<h2 class="STYLE3">标题设置效果</h2>
```





图 1.8　CSS 代码添加注释

1.3　CSS 特性

层叠和继承是 CSS 样式两个最基本的特性，下面分别进行说明。

1.3.1　CSS 层叠性



所谓层叠性，就是 CSS 样式允许重复声明，可以相互覆盖。对于相同 CSS 样式来说，不同位置



Note

的样式其优先级是不同的，主要体现在以下几个方面。

☑　一般来说，行内样式优先于内部样式或者外部样式。

☑　附加了!important 关键字的声明会拥有最高的优先级。注意，必须把!important 命令放置在声明语句与分号之间，否则无效。

☑　对于相同权重的内部样式或外部样式，越接近对象的样式，优先级越高。

☑　基本选择器都拥有一个优先级加权值，说明如下。

- 标签选择器：优先级加权值为 1。
- 伪元素或伪对象选择器：优先级加权值为 1。
- 类选择器：优先级加权值为 10。
- 属性选择器：优先级加权值为 10。
- ID 选择器：优先级加权值为 100。
- 其他选择器：优先级加权值为 0，如通配选择器等。

☑　组合选择器的优先级加权值是组合的基本选择器的加权值之和。以此类推，最后根据加权值之和来决定哪个样式的优先级大。计算的规则如下。

- 统计选择器中 ID 选择器的个数，然后乘以 100。
- 统计选择器中类选择器的个数，然后乘以 10。
- 统计选择器中的标签选择器的个数，然后乘以 1。

【示例 1】本示例演示了如何计算不同样式的加权值。

```
h3{color:#ff7300;}                              /* 加权值=1 分 */
.f14{font-size:14px;}                           /* 加权值=10 分 */
#head{width:960px;}                             /* 加权值=100 分 */
h3 .f14{font-weight:bold;}                      /* 加权值=1 分+10 分=11 分 */
#head h2{border:1px solid #ff73;}               /* 加权值=100 分+1 分=101 分 */
div p{padding:0 10px;}                          /* 加权值=1 分+1 分=2 分 */
div #head{margin:0 auto;}                       /* 加权值=1 分+100 分=101 分 */
#head h2 span{float:right;}                     /* 加权值=100 分+1 分+1 分=102 分 */
#head .f14 em{float:right;}                     /* 加权值=100 分+10 分+1 分=111 分 */
#head .f14 span em{float:right;}                /* 加权值=100 分+10 分+1 分+1 分=112 分 */
#head div h2 .f12 span em{color:#000;}          /* 加权值=100 分+1 分+1 分+10 分+1 分+1 分=114 分 */
```

【示例 2】在本示例中，通过内部样式为同一个元素使用不同的复合选择器，从而为其设置样式属性，通过层叠规则进行比较得出最终样式属性值。页面效果如图 1.9 所示。

```
<style type="text/css">
div{
    margin:0 auto;                              /* 标准浏览器下居中 */
    text-align:center;                          /* IE 怪异模式下居中 */
    }
```






```
.Cent{
    width:400px;                          /* 设置宽度，否则居中看不见效果 */
    border:1px dashed #CC0099;            /* 类别选择器设置边框线 */
    padding:10px 15px;                    /* 设置间距 */
    }
#imp{border:1px dashed #3366FF;}          /* ID 选择器设置边框线 */
.Cent{font-size:14px;}                    /* 类别选择器设置字体大小 */
.Cent p{
    font-size:16px;                       /* 类别选择器和标记选择器一起设置字体大小 */
    font-weight:bold;                     /* 字体加粗 */
    }
.Cent .duanluo{
    font-weight:normal;                   /* 2 次类别选择器设置取消加粗效果 */
    line-height:1.5em;                    /* 段落行高 */
    text-align:left                       /* 文本左对齐 */
    }
.Cent .duanluo span{color:#009966;}       /* 复合选择器设置字体颜色 */
#imp span{
    color:#669933;                        /* ID 选择器和标签选择器进行定义 */
    font-weight:bold;                     /* 字体加粗 */
    font-size:22px                        /* 字体 22 像素，要比较的地方 */
    }
span{font-size:30px !important;           /* <span>标签使用优先级最高的!important 命令*/
    }
span{font-size:40px;!important            /* 错误手写!important 命令的位置 */
    }
</style>

<div class="Cent" id="imp">
<p class="duanluo" id="DL"><span>CSS</span>（Cascading Style Sheet，可译为“层叠样式表”或“级联样式表”)是一组格式设置规则，用于控制 Web 页面的外观。通过使用 CSS 样式设置页面的格式，可将页面的内容与表现形式分离。页面内容存放在HTML 文档中，而用于定义表现形式的CSS 规则则存放在另一个文件中或HTML 文档的某一部分，通常为文件头部分。将内容与表现形式分离，不仅可使维护站点的外观更加容易，而且还可以使 HTML 文档代码更加简练，缩短浏览器的加载时间。
</p>
</div>
```





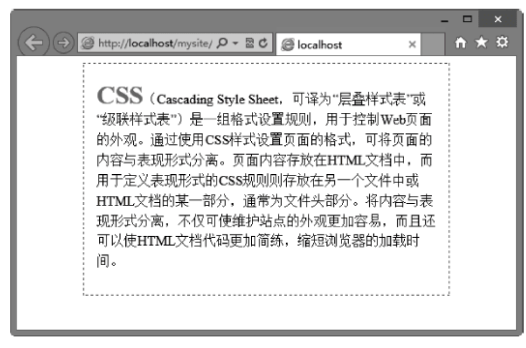


图 1.9　层叠特性测试

在上面的示例中，查看浏览器效果并逐步分析代码，需要注意的是，下面每一步测试时，后面的代码需要删除，故浏览器有多次显示结果，每一步都需要进行浏览器显示，以查看结果。

第 1 步，首先实现浏览器居中，针对<div>标签设置火狐浏览器下居中属性 margin:0 auto；IE 浏览器下居中属性 text-align:center。

```
div{margin:0 auto; text-align:center;}
```

第 2 步，Cent 层设置宽度为 400 像素，如果没有宽度设置，则浏览器上的居中也将无效。接着设置 4 个方向的内间距，最后设置 1 像素颜色为粉红色虚线边框线。

```
.Cent{width:400px; border:1px dashed #CC0099; padding:10px 15px;}
```

第 3 步，通过 ID 值引用 Cent 层，定义 1 像素颜色为粉蓝色虚线边框线，根据前面介绍的层叠规则：类选择器 10 分、ID 选择器 100 分，最终边框线颜色为蓝色。如果将类别选择器 Cent 层和 ID 选择器#imp 定义的顺序颠倒过来，最终结果依然是蓝色，其原因在于 ID 选择器的优先级别高于类选择器。

```
.Cent{width:400px; border:1px dashed #CC0099; padding:10px 15px;}
#imp{border:1px dashed #3366FF;}
```

第 4 步，Cent{}定义字体大小为 14 像素，而.Cent p{}定义字体大小为 16 像素。根据前面介绍的层叠规则：类选择器 10 分、标签选择器 1 分，那么.Cent{}为 10 分、.Cent p{}=10 分+1 分=11 分，故最终结果为段落字体大小为 16 像素且字体加粗显示。

```
.Cent{font-size:14px;}
.Cent p{font-size:16px; font-weight:bold;}
```

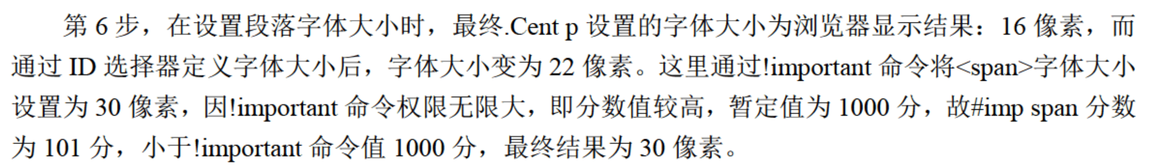
第 5 步，Cent 层中段落添加 class 名 duanluo，定义字体不再加粗显示，行高 1.5em，文本左对齐。对于上一步的加粗设置，如果字体大小无效，则查看加粗结果。行高设置使用相对单位可避免字体大小的改变而影响原先段落文字之间的距离。在使用浏览器居中时，IE 浏览器居中需要设置 text-align:center;，这里更改 Cent 层内文本对齐方式为左对齐，而 Cent 层依然居中显示。

段落内的<span>标签设置字体颜色为#009966，而通过 ID 值设置字体颜色为#669933。根据前面介绍的层叠规则：类选择器 10 分、标签选择器 1 分、ID 选择器 100 分，故.Cent .duanluo span 得分





为 10 分+10 分+1 分=21 分，而#imp span 得分为 100 分+1 分=101 分，最终字体颜色为#669933。



```
.Cent .duanluo{font-weight:normal; line-height:1.5em; text-align:left}
.Cent .duanluo span{color:#009966;}
#imp span{color:#669933; font-weight:bold; font-size:22px}
```

第 6 步，在设置段落字体大小时，最终.Cent p 设置的字体大小为浏览器显示结果：16 像素，而通过 ID 选择器定义字体大小后，字体大小变为 22 像素。这里通过!important 命令将<span>字体大小设置为 30 像素，因!important 命令权限无限大，即分数值较高，暂定值为 1000 分，故#imp span 分数为 101 分，小于!important 命令值 1000 分，最终结果为 30 像素。

若 span{font-size:30px !important;}和#imp span{font-size:50px !important;}比较，根据前面介绍的层叠规则：ID 选择器 100 分、标签选择器 1 分、!important 命令值 1000 分，故 span{}得分为 1000 分（内部属性中 !important）+1 分（标签选择器）=1001 分，而#imp span{font-size:50px !important;}得分为 1000 分（内部属性中 !important）+100 分（ID 选择器）+1 分（标签选择器）=1101 分。



1.3.2　CSS 继承性

所谓继承性，就是指被包含的元素将拥有外层元素的样式效果。继承性最典型的应用就是定义网页标签的默认样式。

【示例】在本示例中，设计 HTML 结构而不书写 CSS 样式，查看页面效果如图 1.10 所示。

```
<div class="head" id="AT">
    <h3><a href="#">More>></a><span>新闻动态</span></h3>
    <div class="list">
        <ul id="S2">
            <li><a href="#">微信“阅读数”是怎么算的？<span>[07-27]</span></a></li>
            <li><a href="#">重振人工智能雄心壮志的时刻已经到了<span>[07-25] </span> </a></li>
            <li><a  href="#">Google  再次“登月”Baseline  工程把基因大数据化<span>  [07-26]</span>
</a></li>
        </ul>
    </div>                <!--list end-->
</div>                        <!--left2-1 end-->
```



图 1.10　HTML 代码页面效果图





Note

在上面的示例中，通过继承的角度考虑标记直接的树形关系，图 1.11 给出继承关系树形图，对比图 1.10，<html>标签是根元素，它是所有 HTML 元素的源头。在每一个分支中，下层是上层的子元素、上层是下层的父元素，故定义 CSS 样式时，编写样式 html, body{font-size:30px}，则页面所有元素都将继承根元素、父元素的字体大小设置，而编写的基本标记选择器、类别选择器、ID 选择器以及复合选择器都是根据 HTML 结构进行编写，尤其是复合选择器中前后标记的位置就是实际 HTML 结构，即图 1.11 所示的树形图。

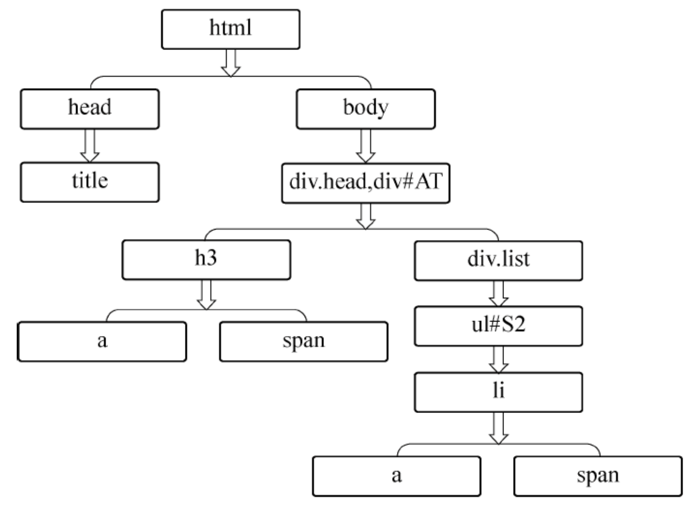


图 1.11　继承关系树形图



1.4　案 例 实 战

本节将学习设计一个完整的页面，体验标准网页的制作过程。案例页面设计效果如图 1.12 所示。



图 1.12　使用 CSS 设计的第一个页面

【操作步骤】

第 1 步，启动 Dreamweaver，新建 HTML 文档，保存为 index.html。





提示：本页面所需要的图片等素材可以参考附赠的源代码。考虑到很多初学者是第一次接触 CSS，本案例稍显复杂，因此建议读者可根据实际情况有选择性学习，或直接跳过本节操作练习。

第 2 步，切换到代码视图，在<body>标签内输入下面代码，构建本页面主体结构，设计本例页面一级框架。

```
<!--[一级框架]-->
<!--顶部-->
<div id="top"></div>
<div id="top1"></div>
<!--主体-->
<div id="main"></div>
<!--底部-->
<div id="footer"></div>
<div id="copyright"></div>
```

在标准布局中，读者应该为每个 div 框架元素定义 id 属性，这些 id 属性如同人的身份证一样，方便 CSS 准确地控制每个 div 布局块。所以，为了阅读和维护的需要，我们应该为它们起一个有意义的名字。

第 3 步，进一步细化页面结构，设计页面内部层次框架。由于本例页面比较简单，嵌套框架不会很深，顶部和底部布局块可能不需要嵌套框架。输入以下完整的 HTML 结构代码。

```
<!--[完整 HTML 框架]-->
<!--顶部-->
<div id="top"></div>
<div id="top1"><img src="images/bg_top.jpg" width="776" height="121" /></div>
<!--主体-->
<div id="main">
    <div id="content">
        <div id="title">Hello World -- 第一个 CSS3+DIV 页面</div>
        <div class="sub">实例</div>
        <div class="box"><div class="tl"><div class="tr"><div class="bl"><div   class="content br">
```

第 4 步，丰富结构内容。使用<pre>标签显示代码内容，使用<a>设计超链接文本，整个页面内容显示如下，代码内容是在网页中居中显示红色字符“Hello World!”。

```
<pre>
&lt;!doctype html&gt;
&lt;html&gt;
    &lt;head&gt;
        &lt;meta charset=&quot;utf-8&quot;&gt;
```




Note

```
            &lt;title&gt;Hello World&lt;/title&gt;
            &lt;style type=&quot;text/css&quot;&gt;
            h1 {
                color: #FF0000;
                text-align: center;
            }
            &lt;/style&gt;
        &lt;/head&gt;
        &lt;body&gt;
            &lt;h1&gt;Hello World! &lt;/h1&gt;
        &lt;/body&gt;
    &lt;/html&gt;
    </pre>
            </div></div></div></div></div>
            <div id="gotop"><a title="跳到页首" href="#top">返回顶部</a></div>
        </div>
    </div>
    <!--底部-->
    <div id="footer"></div>
    <div id="copyright">
        &copy;2017<a href="#" target="_black" >mysite.cn</a> all rights reserved
    </div>
```

上面所用的 HTML 框架代码嵌套层次只有 3 层，其中为了实现圆角区域的显示效果而单独嵌套多层 div 元素除外。

第 5 步，按 Ctrl+S 快捷键保存文档，按 F12 键在浏览器中预览，显示效果如图 1.13 所示，现在还没有定义 CSS 代码，所以看到的还不是最终效果。



图 1.13　页面的 HTML 结构预览效果

第 6 步，在一个单独的文件中编写 CSS。新建 CSS 文档，保存为 style.css，文件扩展名为.css。

第 7 步，（不急于编写 CSS 代码）打开 index.html 文档，在<head>标签内插入一个<link>标签，



输入下面代码，导入上一步新建的外部样式表文件。

```
<!--[在网页中链接外部样式表文件]-->
<LINK href="images/style.css" type=text/css rel=stylesheet>
```

第 8 步，打开 style.css 文档，在其中输入下面的 CSS 代码。

```
/* 公共属性
------------------------------------ */
html { min-width: 776px; }
/* 页面属性：边距为 0，字体颜色为黑色，字体大小为 14 像素，行高为字体大小的 1.6 倍，居中对齐，背景色为天蓝色，字体为宋体等 */
body { margin: 0px; padding: 0px; border: 0px; color: #000; font-size: 14px; line-height: 160%; text-align: center; background: #6D89DD; font-family: '宋体','新宋体',arial,verdana,sans-serif; }
/* 超链接属性：无边距、无边框、无下划线，然后定义正常状态下的颜色、访问过的颜色和鼠标经过时的颜色并显示下划线  */
a { margin: 0px; padding: 0px; border: 0px; text-decoration: none; }
a:link { color: #E66133; }
a:visited{ color: #E66133; }
a:hover{ color: #637DBC; text-decoration: underline; }
/* 预定义格式属性：无首行缩进，浅灰色背景，内边距大小为 0，外边距为 0，字体颜色为蓝色  */
pre { text-indent: 0; background: #DDDDDD; padding: 0; margin: 0; color: blue; }
/* 顶部布局
------------------------------------ */
#top{ width: 776px; margin-right: auto; margin-left: auto; padding: 0px; height: 12px; background: url(images/bg_top1.gif) #fff repeat-x left top; overflow: hidden; }

#top1{ width: 776px; margin-right: auto; margin-left: auto; padding: 0px; height: 121px; }
/*  主体布局
------------------------------------ */
/* 外层定义背景图像，实现麻点显示效果 */
#main{ width: 776px; margin-right: auto; margin-left: auto; padding: 1.2em 0px; background: url(images/bg_dot1.gif) #fff repeat left top; text-align: left; }
/* 内层定义背景颜色为白色，实现中间内容区域遮盖麻点显示 */
#content{ width: 710px; margin-right: auto; margin-left: auto; padding: 1em; background: #fff; }
/* 大标题区域属性 */
#title { font-weight: bold; margin: 0px 0px 0.5em 0px; padding: 0.5em 0px 0.5em 1em; font-size: 24px; color: #00A06B; text-align: left; border-bottom: solid  #9EA3C1  2px; }
/* 小标题区域属性 */
.sub { color: #00A06B; font-weight: bold; font-size: 13px; text-align: left; padding: 1em 2em 0; background:
```



Note

```
url(images/0.gif) #fff no-repeat 1em 74%; }
    /* 内容区域显示属性 */
    .content { text-indent: 2em; font-size: 13px; margin-left: 2em; padding: 1em 6px; }
    /* 页内链接区域属性 */
    #gotop{ width: 100%; margin: 0px; padding: 0px; background: #fff; height: 2em; font-size: 12px; text-align:
right; }
    /* 底部布局
    ----------------------------------- */
    /* 页脚装修图 */
    #footer{ clear: both; width: 776px; margin-right: auto; margin-left: auto; padding: 0px; background:
url(images/bg_bottom.gif) #fff repeat left top; text-align: center; height: 39px; color: #ddd; }
    /* 版权信息 */
    #copyright{ width: 776px; margin-right: auto; margin-left: auto; padding: 5px 0px 0px 0px; background: #fff;
text-align: center; height: 60px; line-height: 13px; font-size: 12px; color: #9EA0BB; }
    #copyright a { color: #667EBE; }
    /* 圆角特效
    ----------------------------------- */
    .box { background: url(images/nt.gif) repeat; }
    .tl { background: url(images/tl.gif) no-repeat top left; }
    .tr { background: url(images/tr.gif) no-repeat top right; }
    .bl { background: url(images/bl.gif) no-repeat bottom left; }
    .br { background: url(images/br.gif) no-repeat bottom right; }
```

读者可能看不懂上面的 CSS 代码，不过没关系，根据上面的提示简单了解即可。其中 width 属性用来定义宽度，background: url(images/bg_bottom.gif) #fff repeat left top;规则用来定义背景图像重复铺展显示，其中 url 指定背景图像的地址，repeat 属性定义铺展显示，left top 表示背景图像的起始位置为左上角。

对于其他属性，上面代码中已有解释，读者可以尝试阅读一下，读不懂也没有关系，毕竟现在仅是开始。相信随着学习的深入，一定会明白上面代码的意思。

另外，本节实例没有使用 CSS3 圆角属性定义区块圆角，而是使用传统方法进行设计，主要考虑初学者的学习门槛，后面章节我们会详细介绍。

第 9 步，按 Ctrl+S 快捷键保存文档，然后在浏览器中再次预览页面，则可以看到最终效果。

1.5 在 线 练 习

本节为读者提供了多个在线练习，以便灵活使用 CSS，强化基本功训练。感兴趣的读者可以扫码练习。



在线练习



使用 CSS3 选择器

CSS3 选择器在 CSS 2.1 选择器的基础上新增了部分属性选择器和伪类选择器，减少对 HTML 类和 ID 的依赖，使编写网页代码更加简单轻松。根据所获取页面中元素的不同，可以把 CSS3 选择器分为 5 大类：元素选择器、关系选择器、伪类选择器、伪对象选择器和属性选择器。本章将详细介绍 CSS3 各类选择器的使用。

【学习要点】

- 正确使用 CSS 基本选择器。
- 灵活使用组合选择器。
- 掌握属性选择器和伪类选择器的应用。





Note

2.1 元素选择器

元素选择器包括标签选择器、类选择器、ID 选择器和通配选择器。

2.1.1 标签选择器

视频讲解

标签选择器也称为类型选择器，直接引用 HTML 标签名称。一般利用标签选择器规定标签的默认样式。

【示例】 本示例演示了如何在文档中定义一个标签样式。通过标签选择器，统一定义网页中段落文本的样式为：字体大小为 12 像素，字体颜色为红色。实现上述默认段落文本效果，可以利用标签选择器定义如下样式。

```
p {
    font-size:12px;              /* 字体大小为 12 像素 */
    color:red;                   /* 字体颜色为红色 */
}
```

在网页设计中，常用标签选择器设计网页默认显示效果，或者统一常用元素的基本样式。标签选择器在 CSS 样式表中容易管理，因为它们都与标签同名。


视频讲解

2.1.2 类选择器

类选择器以一个点（.）前缀开头，然后跟随一个自定义的类名。类选择器能够为网页对象定义不同的样式，实现不同元素拥有相同的样式，相同元素的不同对象拥有不同的样式。

应用类样式可以使用 class 属性来实现，HTML 所有元素都支持该属性，只要在标签中定义 class 属性，然后把该属性值设置为事先定义好的类选择器的名称即可。

【示例 1】 本示例完整演示了如何使用类样式设计段落文本效果。利用类选择器为页面中 3 个相邻的段落文本对象定义不同的样式，其中第 1 和第 3 段文本的字体大小为 12 像素、颜色为红色，第 2 段文本的字体大小为 18 像素、颜色为红色。

第 1 步，启动 Dreamweaver，新建一个网页，在<body>标签内输入以下 3 段段落文本。

```
<p>问君能有几多愁，恰似一江春水向东流。</p>
<p>剪不断，理还乱，是离愁。别是一般滋味在心头。</p>
<p>独自莫凭栏，无限江山，别时容易见时难。流水落花春去也，天上人间。</p>
```

第 2 步，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表。

第 3 步，通过标签选择器将所有段落文本的字体大小定义为 12 像素，字体颜色定义为红色，代码如下。





```
p {
    font-size:12px;      /* 字体大小为 12 像素 */
    color:red;           /* 字体颜色为红色 */
}
```

第 4 步，如果仅定义第 2 段文本的字体大小为 18 像素，这时就可以使用类选择器。假设定义一个 18 像素大小的字体类，代码如下。

```
.font18px { font-size:18px;}
```

第 5 步，在第 2 段段落标签中引用 font18px 类样式，代码如下。

```
<p>问君能有几多愁，恰似一江春水向东流。</p>
<p class="font18px">剪不断，理还乱，是离愁。别是一般滋味在心头。</p>
<p>独自莫凭栏，无限江山，别时容易见时难。流水落花春去也，天上人间。</p>
```

第 6 步，在浏览器中预览的效果如图 2.1 所示，此时可以看到 3 段文本的显示样式，其中第 2 段文本被单独放大显示。

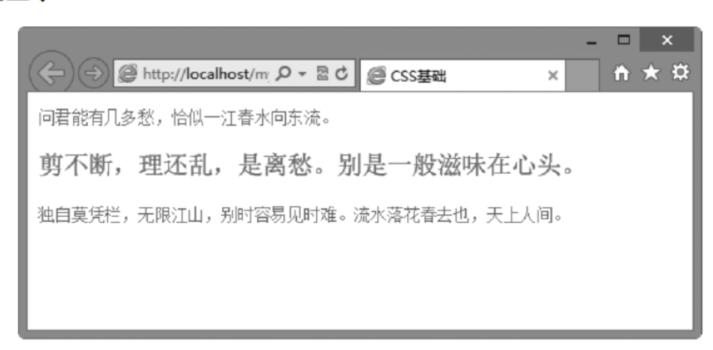


图 2.1　类选择器应用效果

【示例 2】本示例演示了如何在对象中应用多个样式类。class 属性可以包含多个类，因此可以设计复合样式类。用户可以按如下步骤上机练习体验。

第 1 步，复制示例 1 中文档，并在内部样式表中定义 3 个类：font18px、underline 和 italic。

第 2 步，在段落文本中分别引用这些类，其中第 2 段文本标签引用了 3 个类，代码如下，演示效果如图 2.2 所示。

```
<style type="text/css">
p {/* 段落默认样式 */
    font-size:12px;                  /* 字体大小为 12 像素 */
    color:red;                       /* 字体颜色为红色 */
    }
.font18px {/* 字体大小类  */
```




Note

```
        font-size:18px;                          /* 字体大小为 18 像素 */
    }
.underline {/* 下划线类 */
        text-decoration:underline;               /* 字体修饰为下划线 */
    }
.italic {/* 斜体类 */
        font-style:italic;                       /* 字体样式为斜体 */
    }
</style>

<p class="underline">问君能有几多愁，恰似一江春水向东流。</p>
<p class="font18px italic underline">剪不断，理还乱，是离愁。别是一般滋味在心头。</p>
<p class="italic">独自莫凭栏，无限江山，别时容易见时难。流水落花春去也，天上人间。</p>
```

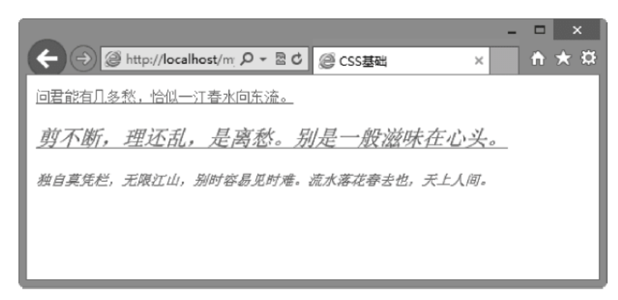


图 2.2　多类引用应用效果


视频讲解

2.1.3　ID 选择器

ID 选择器以井号（#）作为前缀，接着是一个自定义的 ID 名。应用 ID 选择器可以使用 id 属性来实现，HTML 所有元素都支持该属性，只要在标签中定义 id 属性，然后把该属性值设置为事先定义好的 ID 选择器的名称即可。

【示例 1】本示例演示了如何在文档中设置 ID 样式。

第 1 步，启动 Dreamweaver，新建一个网页，在<body>标签内输入<div>标签。

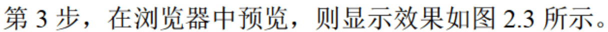
```
<div id="box">问君能有几多愁，恰似一江春水向东流。</div>
```

第 2 步，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表。然后为该盒子定义固定宽和高，并设置背景图像，以及边框和内边距大小，代码如下。

```
#box {/* ID 样式  */
        background:url(images/2.jpg) center bottom;  /* 定义背景图像并居中、底部对齐 */
        height:200px;                                /* 固定盒子的高度 */
        width:400px;                                 /* 固定盒子的宽度 */
```



```
        border:solid 2px red;                    /* 边框样式 */
        padding:100px;                           /* 增加内边距 */
    }
```



第 3 步，在浏览器中预览，则显示效果如图 2.3 所示。



图 2.3　ID 选择器的应用效果

也可以为 ID 选择器指定标签范围。采用这种方法能够提高该样式的优先级。

【示例 2】针对上面示例，在 ID 选择器前面增加一个<div>标签，这样 div#box 选择器的优先级会大于#box 选择器的优先级。在同等条件下，浏览器会优先解析 div#box 选择器定义的样式，代码如下。

```
<style type="text/css">
div#box {/* ID 样式  */
    background:url(images/bg1.gif) center bottom;  /* 定义背景图像并居中、底部对齐 */
    height:200px;                                  /* 固定盒子的高度 */
    width:400px;                                   /* 固定盒子的宽度 */
    border:solid 2px red;                          /* 边框样式 */
    padding:20px;                                  /* 增加内边距 */
    }
</style>

<div id="box">问君能有几多愁，恰似一江春水向东流。</div>
```

注意，也可以使用这种方式限制类选择器的应用范围，并增加其优先级。

2.1.4　通配选择器



如果 HTML 所有元素都需要定义相同的样式，这时不妨使用通配选择器。通配选择器是固定的，



它使用星号（*）来表示。

【示例】针对上面示例中清除边距样式，可以使用下面方式来定义。



```
* {
    margin: 0;
    padding: 0;
}
```

2.2 关系选择器

把两个基本选择器组合在一起，就形成了一个复杂的关系选择器，通过关系选择器可以精确匹配页面元素。



2.2.1 包含选择器

包含选择器通过空格标识符来表示，前面的一个选择器表示包含框对象的选择器，而后面的选择器表示被包含的选择器，如图 2.4 所示。

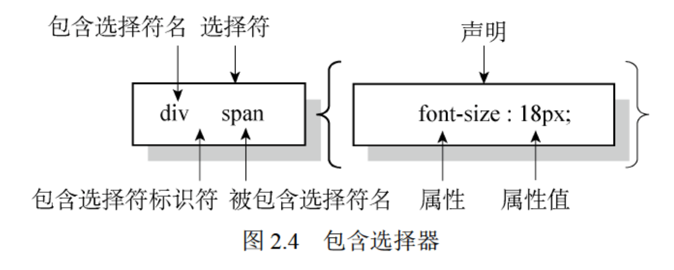


图 2.4　包含选择器

【示例】本示例演示了如何使用包含选择器为不同层次下的标签定义样式。

启动 Dreamweaver，新建一个网页，在<body>标签内输入如下结构。

```
<div id="wrap">
    <div id="header">
        <p>头部区域第 1 段文本</p>
        <p>头部区域第 2 段文本</p>
        <p>头部区域第 3 段文本</p>
    </div>
    <div id="main">
        <p>主体区域第 1 段文本</p>
        <p>主体区域第 2 段文本</p>
        <p>主体区域第 3 段文本</p>
    </div>
</div>
```





在<head>标签内添加<style type="text/css">标签，定义一个内部样式表。然后定义样式，希望实现如下设计目标。

☑　定义<div id="header">包含框内的段落文本字体大小为 14 像素。

☑　定义<div id="main">包含框内的段落文本字体大小为 12 像素。

这时可以利用包含选择器来快速定义它们的样式，代码如下。

```
#header p { font-size:14px;}
#main p {font-size:12px;}
```

2.2.2　子选择器



子选择器使用尖角号（>）表示，子选择器是指定父元素所包含的子元素，如图 2.5 所示。

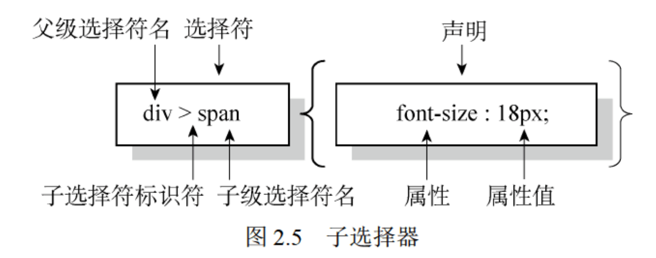


图 2.5　子选择器

【示例】本示例演示了如何使用子选择器为不同结构中的标签定义样式。

启动 Dreamweaver，新建一个网页，在<body>标签内输入如下结构。

```
<h2>
    <span>HTML 文档树状结构</span>
</h2>
<div id="box">
    <span class="font24px">问君能有几多愁，恰似一江春水向东流。</span>
</div>
```

在<head>标签内添加<style type="text/css">标签，定义一个内部样式表。然后定义所有 span 元素的字体大小为 12 像素，再利用子选择器来定义所有 div 元素包含的子元素 span 的样式为 24 像素，代码如下。

```
span {/* span 元素的默认样式 */
    font-size:12px;    /* 字体大小为 12 像素 */
}
div > span {/* div 元素包含的 span 子元素的默认样式 */
    font-size:24px;    /* 字体大小为 24 像素 */
}
```

在浏览器中预览，显示效果如图 2.6 所示。





Note

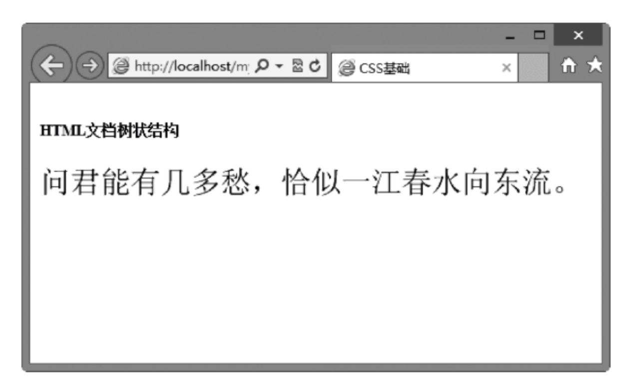


图 2.6　子选择器应用效果

从图 2.6 可以看到，包含在 div 元素内的子元素 span 被定义了字体大小为 24 像素。通过这种方式可以准确定义某个或一组子元素的样式，而不再为它们定义 ID 属性或者 Class 属性。



视频讲解

2.2.3　相邻选择器

相邻选择器通过加号（+）分隔符进行定义。其基本结构是第 1 个选择器指定前面相邻元素，后面的选择器指定后面相邻元素，如图 2.7 所示。前后选择符的关系是兄弟关系，即在 HTML 结构中，两个标签前为“兄”后为“弟”，否则样式无法应用。

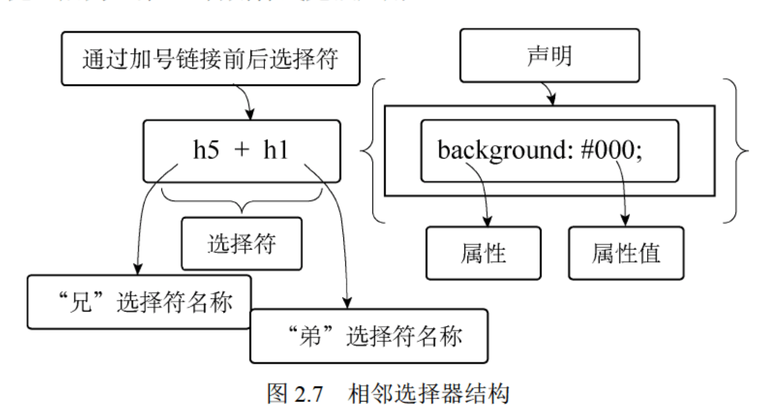


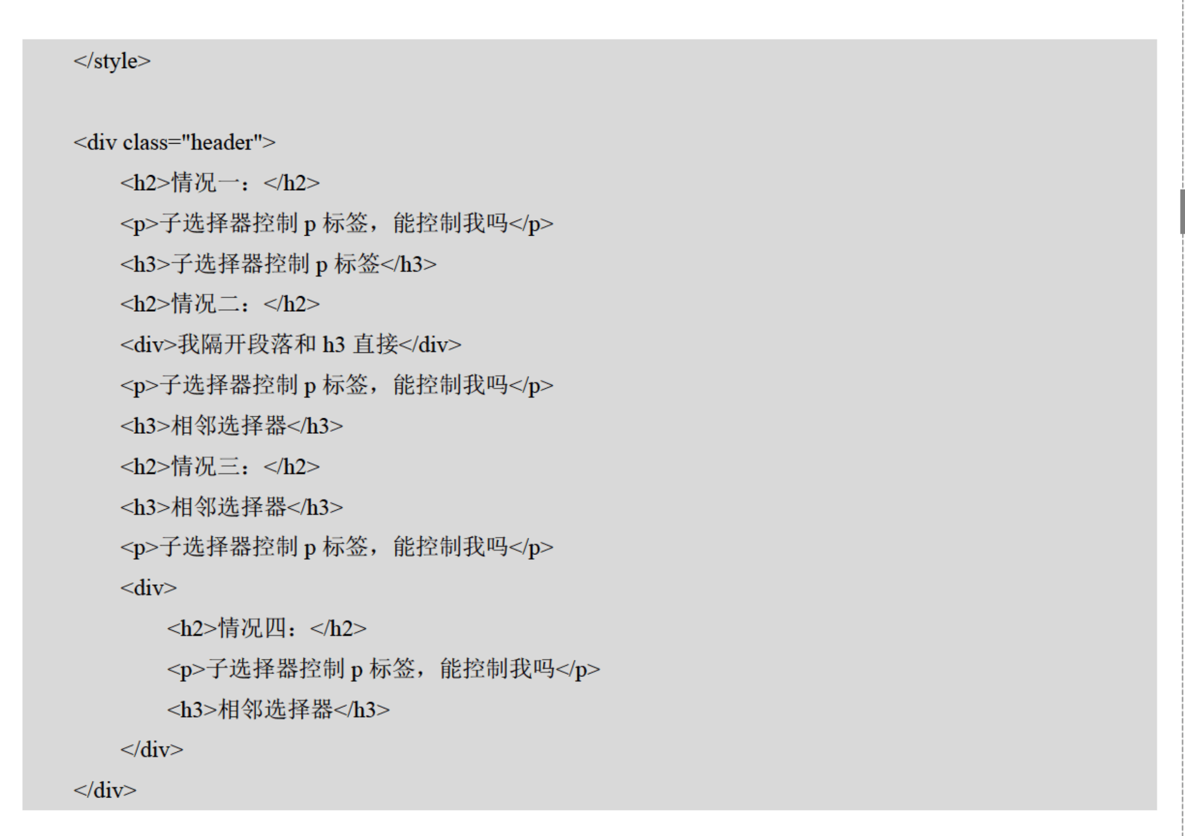
图 2.7　相邻选择器结构

【示例】 在本示例中，通过 4 种情况测试相邻选择符的应用范围，代码如下。

```
<style type="text/css">
h2, p, h3 {
    margin: 0;                              /* 清除默认边距 */
    padding: 0;                             /* 清除默认间距 */
    height: 30px;                           /* 初始化设置高度为 30 像素 */
    }
p+h3 { background-color: #0099FF;           /* 设置背景色 */
    }
```







```
</style>

<div class="header">
    <h2>情况一：</h2>
    <p>子选择器控制 p 标签，能控制我吗</p>
    <h3>子选择器控制 p 标签</h3>
    <h2>情况二：</h2>
    <div>我隔开段落和 h3 直接</div>
    <p>子选择器控制 p 标签，能控制我吗</p>
    <h3>相邻选择器</h3>
    <h2>情况三：</h2>
    <h3>相邻选择器</h3>
    <p>子选择器控制 p 标签，能控制我吗</p>
    <div>
        <h2>情况四：</h2>
        <p>子选择器控制 p 标签，能控制我吗</p>
        <h3>相邻选择器</h3>
    </div>
</div>
```

在浏览器中预览，页面效果如图 2.8 所示。

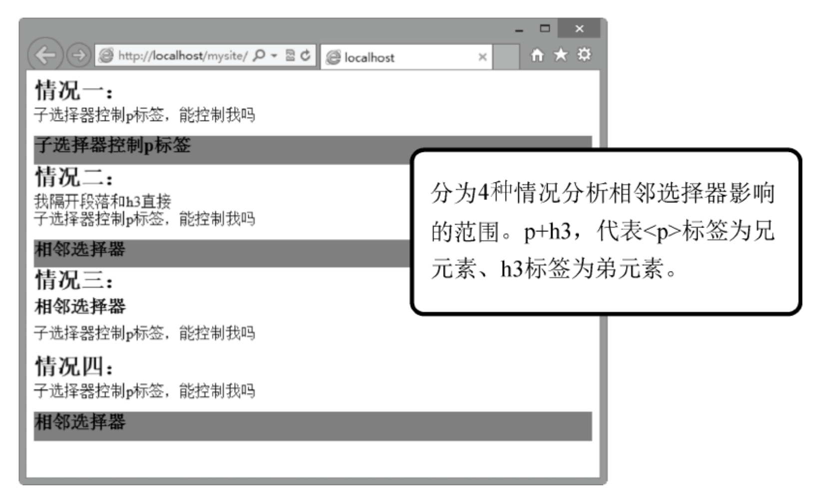


图 2.8　相邻选择器

在上面示例中，将相邻选择器分成 4 种情况进行分析。

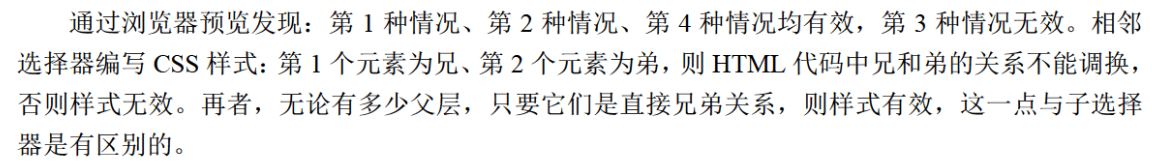
第 1 种：正常情况下，<p>标签和<h3>标签是兄弟元素；

第 2 种：添加一个<div>标签，将<p>标签和<h3>标签与第 1 种情况进行间隔，测试在有元素间隔时，样式是否有效；




第 3 种：<h3>标签为兄元素、<p>标签为弟元素，测试是否受影响；

第 4 种：为<p>标签和<h3>标签添加一个父层，查看是否受影响；



通过浏览器预览发现：第 1 种情况、第 2 种情况、第 4 种情况均有效，第 3 种情况无效。相邻选择器编写 CSS 样式：第 1 个元素为兄、第 2 个元素为弟，则 HTML 代码中兄和弟的关系不能调换，否则样式无效。再者，无论有多少父层，只要它们是直接兄弟关系，则样式有效，这一点与子选择器是有区别的。

Note



视 频 讲 解

2.2.4　兄弟选择器

CSS3 增加了一种新的选择器组合形式——兄弟选择器。它通过波浪（~）分隔符进行定义。其基本结构是第 1 个选择器指定同级前置元素，后面的选择器指定其后同级所有匹配元素，如图 2.9 所示。前后选择符的关系是兄弟关系，即在 HTML 结构中两个标签前为“兄”后为“弟”，否则样式无法应用。

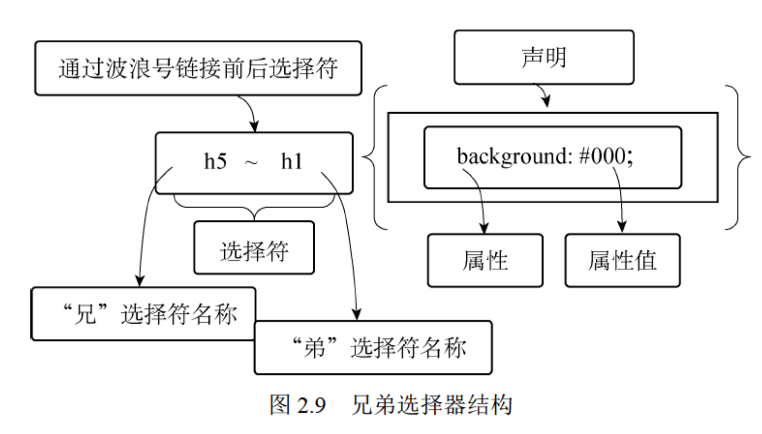


图 2.9　兄弟选择器结构

兄弟选择器能够选择前置元素后同级的所有匹配元素，而相邻选择器只能选择前置元素后相邻的一个匹配元素。

【示例】以上节示例为基础，修改其中的 p+h3 { background-color: #0099FF; }样式为 p ~ h3 { background-color: #0099FF; }，具体样式代码如下。

```
h2, p, h3 {
    margin: 0;                          /* 清除默认边距 */
    padding: 0;                         /* 清除默认间距 */
    height: 30px;                       /* 初始化设置高度为 30 像素 */
    }
p ~ h3 { background-color: #0099FF;}    /* 设置背景色 */
```

在浏览器中预览，则页面效果如图 2.10 所示。可以看到在<div class="header">包含框中，位于<p>标签后的所有<h3>标签都被选中，设置背景色为蓝色。





图 2.10　兄弟选择器



2.2.5　分组选择器



分组选择器通过逗号（,）分隔符进行定义。其基本结构是第一个选择器指定匹配元素，后面的选择器指定另一个匹配元素，最后把前后选择器匹配的元素都选取，如图 2.11 所示。

通过分组选择器可以实现集体声明，将样式表中一致的 CSS 样式放在一起，接着将这些选择器用逗号链接，以减少代码的书写量。

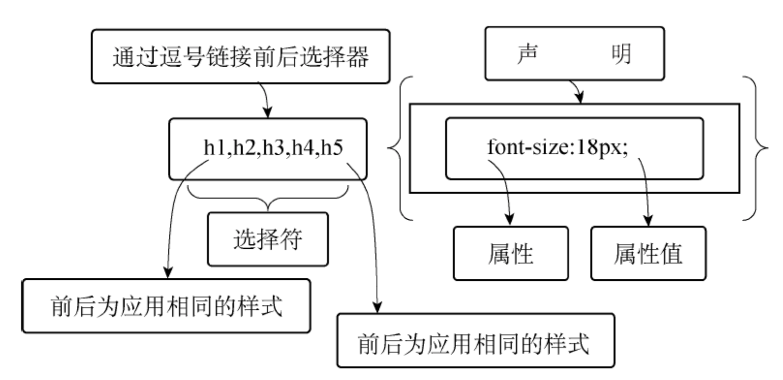


图 2.11　分组选择器

【示例】在本示例中，将上面学过的复合选择器通过分组选择器集中声明，代码如下。

```
<style type="text/css">
h1, h2, h3, h4, h5, h6 {
    background-color: #99CC33;              /* 设置背景色 */
    margin: 0;                              /* 清除标题的默认外边距 */
    margin-bottom: 10px;                    /* 使用下边距拉开各个标题之间的距离 */
    }
h1+h2, h2+h3, h4+h5 { color: #0099FF;}     /* 兄弟关系设置字体的颜色 */
```





Note

```
body>h6, h1>span, h4>span { font-size: 20px;}                /* 子选择器设置字体的大小 */
h2 span, h3 span { padding: 0 20px;}                         /* 设置<span>标签的左右间距 */
h5 span[class], h6 span[class] { background-color: #CC0033;} /* 为h5、h6 标题中含有 class 属性的<span>标签
                                                                设置背景色 */
</style>

<h1>h1 元素<span>这里是 span 元素</span></h1>
<h2>h2 元素<span>这里是 span 元素</span></h2>
<h3>h3 元素<span>这里是 span 元素</span></h3>
<h4>h4 元素<span>这里是 span 元素</span></h4>
<h5>h5 元素<span class="S1">这里是 span 元素</span></h5>
<h6>h6 元素<span class="S2">这里是 span 元素</span></h6>
```

在浏览器中预览，页面效果如图 2.12 所示。



图 2.12　分组选择器

在上面示例中，将<h1>~<h6>标签设置背景色为#99CC33，清除默认边距，通过下边距区分标题标签的空间。h1+h2,h2+h3,h4+h5 代表 3 种兄弟元素均设置字体颜色为#0099FF；body>h6, h1>span,h4>span 代表<body>标签的子元素<h6>、<h1>标签的子元素<span>、<h4>标签的子元素<span>这 3 种情况下的字体大小为 20 像素；h2 span,h3 span 代表<h2>标签内的<span>、<h3>标签内的<span>（子孙辈皆可）左右边距为 20 像素；h5 span[class],h6 span[class]代表<h5>标签中的<span>标签含有 Class 属性、<h6>标签中的<span>标签含有 class 属性，背景色为#CC0033。



2.3　属性选择器

CSS3 在 CSS2 基础上新增加了 3 个属性选择器，如 E[attr^="value"]、E[attr$="value"]和 E[attr*="value"]。这 7 个属性选择器具体说明如下。

☑　E[attr]：只使用属性名，但没有确定任何属性值。







- ☑ E[attr="value"]：指定属性名，并指定了该属性的属性值。
- ☑ E[attr~="value"]：指定属性名，并且具有属性值，此属性值是一个词列表，并且以空格隔开，其中词列表中包含了一个 value 词，而且等号前面的~不能不写。
- ☑ E[attr^="value"]：指定属性名，并且有属性值，属性值是以 value 开头的。
- ☑ E[attr$="value"]：指定属性名，并且有属性值，属性值是以 value 结束的。
- ☑ E[attr*="value"]：指定属性名，并且有属性值，而且属值中包含了 value。
- ☑ E[attr|="value"]：指定属性名，并且属性值是 value 或者以 value-开头的值（如 zh-cn）。

提示： 在上面的语法形式中，E 表示匹配元素的选择符，可以省略；中括号为属性选择器标识符，不可或缺；attr 表示 HTML 属性名；value 表示 HTML 属性值或者 HTML 属性值包含的子字符串。

目前，主流浏览器都支持属性选择器，虽然早期 IE 浏览器存在不兼容问题，如 IE 6，但不影响属性选择器的普及和使用。

【示例】为了更好地演示 CSS3 属性选择器的应用，本示例将设计一个简单的图片灯箱导航示例，其中 HTML 结构如下。

```
<div class="pic_box">
    <img src="images/bg1.jpg" />
    <div class="nav">
        <a href="#1" class="links item first" title="w3cplus" target="_blank" id="first" >1</a>
        <a href="#2" class="links active item" title="test website" target="_blank" lang="zh">2</a>
        <a href="#3" class="links item" title="this is a link" lang="zh-cn">3</a>
        <a href="#4" class="links item" target="_balnk" lang="zh-tw">4</a>
        <a href="#5" class="links item" title="zh-cn">5</a>
        <a href="#6" class="links item" title="website link" lang="zh">6</a>
        <a href="#7" class="links item" title="open the website" lang="cn">7</a>
        <a href="#8" class="links item" title="close the website" lang="en-zh">8</a>
        <a href="#9" class="links item" title="http://www.baidu.com">9</a>
        <a href="#10" class="links item last" id="last">10</a>
    </div>
</div>
```

使用 CSS 适当美化该结构块，具体代码如下所示，预览效果如图 2.13 所示。

```
<style type="text/css">
/*灯箱外框样式*/
.pic_box { border: solid 6px #bbb; position: relative; float: left; }
.pic_box img { border: solid 1px red; }
/*导航框样式*/
```



Note

```
    .nav { background: #fff; border: 1px solid #aaa; padding: 6px 12px; float: left; position: absolute; bottom: 6px; right: 12px; }
    /*导航按钮样式*/
    .nav a { float: left; display: block; height: 20px; line-height: 20px; width: 20px; -moz-border-radius: 10px; -webkit-border-radius: 10px; border-radius: 10px; text-align: center; background: #f00; color: #fff; margin-right: 5px; text-decoration: none; }
.nav a:hover { background: green; }
    </style>
```



图 2.13　设计的灯箱广告效果图

下面结合这个示例，具体分析每个属性选择器的使用方法。

1. E[attr]

E[attr]属性选择器可以选择包含指定属性的元素，例如：

```
.nav a[id] {background: blue; color:yellow;font-weight:bold;}
```

上面代码表示的是，选择了 div.nav 下所有带有 id 属性的 a 元素，并在这个元素上使用背景色为蓝色，前景色为黄色，字体加粗的样式。对照上面的 HTML 结构不难发现，只有第一个和最后一个链接使用了 id 属性，所以选中了这两个 a 元素，效果如图 2.14 所示。

也可以指定多属性：

```
.nav a[href][title] {background: yellow; color:green;}
```

上面代码表示的是，选择 div.nav 下的同时具有 href 和 title 两个属性的 a 元素，效果如图 2.15 所示。





图 2.14　属性快速匹配



图 2.15　多属性快速匹配



2. E[attr="value"]

E[attr="value"]选择器能精确选择需要的元素，例如：

```
.nav a[id="first"] {background: blue; color:yellow;font-weight:bold;}
```

选中 div.nav 中的 a 元素，并且这个元素有一个 id="first"属性值，则预览效果如图 2.16 所示。

E[attr="value"]属性选择器也可以多个属性并写，进一步缩小选择范围，用法如下所示，则预览效果如图 2.17 所示。

```
.nav a[href="#1"][title] {background: yellow; color:green;}
```



图 2.16　属性值快速匹配



图 2.17　多属性值快速匹配

3. E[attr~="value"]

E[attr~="value"]属性选择器匹配一个或多个词列表，如果是列表，则需要用空格隔开，只要属性值中有一个 value 相匹配就可以选中该元素。例如：

```
.nav a[title~="website"]{background:orange;color:green;}
```

在 div.nav 下的 a 元素的 title 属性中，只要其属性值中含有 website 这个词就会被选择，结果 a 元素中“2”“6”“7”“8”这 4 个 a 元素的 title 中都含有，所以被选中，如图 2.18 所示。

4. E[attr^="value"]

E[attr^="value"]属性选择器选择 attr 属性值以 value 开头的所有元素，例如：

```
.nav a[title^="http://"]{background:orange;color:green;}
.nav a[title^="mailto:"]{background:green;color:orange;}
```

上面代码表示的是选择了 title 属性，并且以"http://"和"mailto:"开头的属性值的所有 a 元素，匹配效果如图 2.19 所示。



图 2.18　属性值局部词匹配

图 2.19　匹配属性值开头字符串的元素

5. E[attr$="value"]

E[attr$="value"]表示选择 attr 属性值以 value 结尾的所有元素，例如：





Note

```
.nav a[href$="png"]{background:orange;color:green;}
```

上面代码表示选择 div.nav 中元素有 href 属性，并以 png 结尾的 a 元素。

6. E[attr*="value"]

E[attr*="value"]属性选择器表示选择 attr 属性值中包含子串"value"的所有元素。例如：

```
.nav a[title*="site"]{background:black;color:white;}
```

上面代码表示选择 div.nav 中 a 元素的 title 属性中只要有"site"字符串的元素。上面样式的预览效果如图 2.20 所示。

7. E[attr|="value"]

E[attr|="value"]选择器会选择 attr 属性值等于 value 或以 value-开头的所有元素，例如：

```
.nav a[lang|="zh"]{background:gray;color:yellow;}
```

上面代码会选中 div.nav 中 lang 属性等于 zh 或以 zh-开头的所有 a 元素，如图 2.21 所示。



图 2.20　匹配属性值中的特定子串



图 2.21　匹配属性值开头字符串的元素

2.4　伪类选择器

伪类选择器包括伪类和伪对象选择器，伪类选择器以冒号（:）作为前缀标识符。冒号前可以添加选择符，限定伪类应用的范围，冒号后为伪类和伪对象名，冒号前后没有空格，如图 2.22 所示。

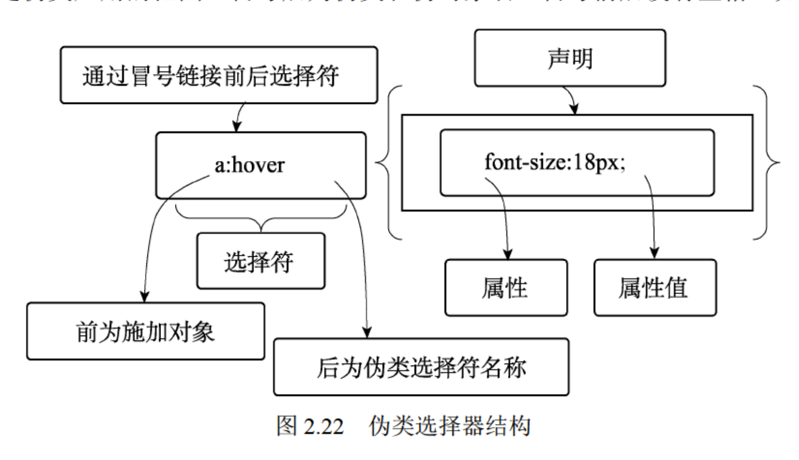


图 2.22　伪类选择器结构

提示：CSS 伪类选择器有以下两种用法。







☑　单纯式

```
E:pseudo-class { property:value}
```

其中 E 为元素，pseudo-class 为伪类名称，property 是 CSS 的属性，value 为 CSS 的属性值。例如：

```
a:link {color:red;}
```

☑　混用式

```
E.class:pseudo-class{property:value}
```

其中.class 表示类选择符。把类选择符与伪类选择符组成一个混合式的选择器，能够设计更复杂的样式，以精准匹配元素。例如：

```
a.selected:hover {color: blue;}
```



视频讲

2.4.1　动态伪类

动态伪类是一类行为类样式，只有当用户与页面进行交互时才有效，包括以下两种形式。

☑　锚点伪类，如:link、:visited。

☑　行为伪类，如:hover、:active 和:focus。

【示例】本示例将使用动态伪类选择器设计一组 3D 动态效果的按钮样式，效果如图 2.23 所示。



图 2.23　设计 3D 按钮样式

【操作步骤】

第 1 步，设计一个 HTML 文档结构。创建一个新的 HTML 文档，并添加一个列表，在列表项中包含基本的锚链接。就这么简单，不需要任何额外的<div>或者<span>标签，也不用添加 id 和 class 属性，一切效果都通过 CSS 进行控制。结构代码如下。

```
<ul id="container">
    <li><a href="#" class="button gray">灰色风格按钮</a></li>
    <li><a href="#" class="button pink">粉红风格按钮</a></li>
    <li><a href="#" class="button blue">蓝色风格按钮</a></li>
    <li><a href="#" class="button green">绿色风格按钮</a></li>
    <li><a href="#" class="button turquoise">天蓝色风格按钮</a></li>
    <li><a href="#" class="button black">黑色风格按钮</a></li>
```





```
        <li><a href="#" class="button darkgray">深灰色风格按钮</a></li>
        <li><a href="#" class="button yellow">黄色风格按钮</a></li>
        <li><a href="#" class="button purple">紫色风格按钮</a></li>
        <li><a href="#" class="button darkblue">深蓝色风格按钮</a></li>
</ul>
```

为了能够演示不同色彩的 CSS 样式，通过列表结构设计一组类似的按钮。给每一个按钮定义一类不同的颜色，通过对比可以发现这类样式设计的优势和便捷之处。

第 2 步，新建内部样式，定义基本的按钮类样式，代码如下。

```
ul { list-style: none; }
a.button {
        display: block; float: left;
        position: relative;
        height: 25px; width: 120px;
        margin: 0 10px 18px 0;
        text-decoration: none;
        font: 12px "Helvetica Neue", Helvetica, Arial, sans-serif;
        font-weight: bold; line-height: 25px; text-align: center;
}
```

第 3 步，为该类样式增加行为样式，让按钮实现动态效果。这里主要使用了:hover 伪类选择器。例如，为灰色系按钮设计鼠标经过时的动态样式效果，主要包括字体颜色和背景色的变化，以及边框线的变换，以模拟立体效果。

```
.gray,.gray:hover {
        color: #555;
        border-bottom: 4px solid #b2b1b1;
        background: #eee;
}
.gray:hover { background: #e2e2e2; }
```

第 4 步，定义双边框样式。通过预览会发现现在的按钮边框显得比较单薄，需要为按钮定义粗边框的底部效果，同时还需要增加一点行间距，因此这里使用了:before 和:after 伪类样式，代码如下。

```
a.button:before, a.button:after {
        content: '';
        position: absolute;
        left: -1px; bottom: -1px;
        height: 25px; width: 120px;
```





```
        border-radius: 3px;
}
a.button:before {
        height: 23px;
        bottom: -4px;
        border-top: 0;
        border-radius: 0 0 3px 3px;
        box-shadow: 0 1px 1px 0px #bfbfbf;
}
```

第 5 步，为了彰显按钮的金属特质，不妨借助 CSS3 的特效定义圆角效果，代码如下。

```
a.button {border-radius: 3px;}
```

第 6 步，为边框定义阴影效果，代码如下。

```
a.button:before,a.button:after { border-radius: 3px;}
a.button:before {
        border-radius: 0 0 3px 3px;
        box-shadow: 0 1px 1px 0px #bfbfbf;
}
```

第 7 步，定义鼠标经过和访问过按钮伪类状态类样式，设计渐变背景色特效，代码如下。

```
/* 设计灰色主题风格*/
a.gray,a.gray:hover,a.gray:visited {
        color: #555;
        border-bottom: 4px solid #b2b1b1;
        text-shadow: 0px 1px 0px #fafafa;
        background: #eee;
        background: linear-gradient(to top,  #eee,  #e2e2e2);
        box-shadow: inset 1px 1px 0 #f5f5f5;
}
.gray:before,.gray:after {
        border: 1px solid #cbcbcb;
        border-bottom: 1px solid #a5a5a5;
}
.gray:hover {
        background: #e2e2e2;
        background: linear-gradient(to top,  #e2e2e2,  #eee);
}
```



第 8 步，利用:active 伪类选择器定义对象激活下的样式效果。



```
/* 激活状态样式 */
a.button:active {
    border: none;
    bottom: -4px;
    margin-bottom: 22px;
    box-shadow:   1px 1px 0 #fff, inset 0 1px 1px rgba(0, 0, 0, 0.3);
}
a.button:active:before,
a.button:active:after {
    border: none;
    box-shadow: none;
}
```



2.4.2 结构伪类

结构伪类是根据文档结构的相互关系来匹配特定的元素，从而减少文档元素的 class 属性和 id 属性的无序设置，使得文档更加简洁。

结构伪类形式多样，但用法固定，以便设计各种特殊样式效果。结构伪类主要包括以下几种。

- ☑ :first-child：第一个子元素。
- ☑ :last-child：最后一个子元素。
- ☑ :nth-child()：按正序匹配特定子元素。
- ☑ :nth-last-child()：按倒序匹配特定子元素。
- ☑ :nth-of-type()：在同类型中匹配特定子元素。
- ☑ :nth-last-of-type()：按倒序在同类型中匹配特定子元素。
- ☑ :first-of-type：第一个同类型子元素。
- ☑ :last-of-type：最后一个同类型子元素。
- ☑ :only-child：唯一子元素。
- ☑ :only-of-type：同类型的唯一子元素。
- ☑ :empty：空元素。

【示例 1】本示例是设计一个排行榜栏目列表样式，效果如图 2.24 所示。在列表框中为每个列表项定义相同的背景图像。







图 2.24　设计排行榜栏目样式

设计的列表结构如下。

```
<div id="wrap">
    <ul id="container">
        <li><a href="#">送君千里 终须一别</a></li>
        <li><a href="#">旅行的意义</a></li>
        <li><a href="#">南师虽去，精神永存</a></li>
        <li><a href="#">榴莲糯米糍</a></li>
        <li><a href="#">阿尔及利亚天命之年</a></li>
        <li><a href="#">白菜鸡肉粉丝包</a></li>
        <li><a href="#">《展望塔上的杀人》</a></li>
        <li><a href="#">我们，只会在路上相遇</a></li>
    </ul>
</div>
```

设计的列表样式如下。

```
/*定位栏目位置*/
#wrap { width: 260px; height: 276px; background: url(images/bg1.jpg) no-repeat;}
/*初始化列表结构样式*/
#wrap ul { list-style-type: none; margin: 0; padding: 0; font-size: 12px; color: #777; }
#wrap li { background: url(images/top10-bullet.png) no-repeat 2px 10px; padding: 1px 0px 0px 28px; line-height:
30px; }
#wrap li a { text-decoration: none; color: #777; }
#wrap li a:hover { color: #F63; }
```





下面结合这个栏目示例，具体分析结构伪类选择器的用法。

1. :first-child

【示例 2】如果设计第一个列表项前的图标为 1，且字体加粗显示，则使用:first-child 匹配，代码如下。

```
#wrap li:first-child {
    background-position:2px 10px;
    font-weight:bold;
}
```

2. :last-child

【示例 3】如果单独给最后一个列表项定义样式，就可以使用:last-child 来匹配。

```
#wrap li:last-child {background-position:2px -277px;}
```

显示效果如图 2.25 所示。

3. :nth-child()

:nth-child()可以选择一个或多个特定的子元素。该函数有多种用法。

```
:nth-child(length);/*参数是具体数字*/
:nth-child(n);/*参数是 n，n 从 0 开始计算*/
:nth-child(n*length)/*n 的倍数选择，n 从 0 开始算*/
:nth-child(n+length);/*选择大于或等于 length 的元素*/
:nth-child(-n+length)/*选择小于或等于 length 的元素*/
:nth-child(n*length+1);/*表示隔几选一*/
```

在 nth-child()函数中，参数 length 为一个整数，n 表示一个从 0 开始的自然数。

:nth-child()可以定义值，值可以是整数，也可以是表达式，用来选择特定的子元素。

【示例 4】下面 6 个样式分别匹配列表中第 2～7 个列表项，并分别定义它们的背景图像 Y 轴坐标位置，显示效果如图 2.26 所示。

```
#wrap li:nth-child(2) { background-position: 2px -31px; }
#wrap li:nth-child(3) { background-position: 2px -72px; }
#wrap li:nth-child(4) { background-position: 2px -113px; }
#wrap li:nth-child(5) { background-position: 2px -154px; }
#wrap li:nth-child(6) { background-position: 2px -195px; }
#wrap li:nth-child(7) { background-position: 2px -236px; }
```







图 2.25　设计最后一个列表项样式



图 2.26　第 2～7 个列表项样式

注意，这种函数参数用法不能引用负值，也就是说，li:nth-child(-3)是不正确的使用方法。

☑　:nth-child(n)

在:nth-child(n)中，n 是一个简单的表达式，其取值从 0 开始计算，到什么时候结束不确定，需结合文档结构而定，如果在实际应用中直接这样使用，将会选中所有子元素。

【示例 5】在上面示例中，如果在 li 中使用:nth-child(n)，那么将选中所有的 li 元素。

```
#wrap li:nth-child(n) {text-decoration:underline;}
```

上述样式类似于：

```
#wrap li {text-decoration:underline;}
```

其实，nth-child(n)是这样计算的：

n=0 表示没有选择元素；

n=1 表示选择第 1 个 li；

n=2 表示选择第 2 个 li。

依此类推，这样就选中了所有的 li。

☑　:nth-child(2n)

【示例 6】:nth-child(2n)是:nth-child(n)的一种变体，使用它可以选择 n 的 2 倍，当然其中 2 可以换成需要的数字，分别表示不同的倍数。

```
#wrap li:nth-child(2n) {font-weight:bold;}
```

等价于：

```
#wrap li:nth-child(even) {font-weight:bold;}
```

预览效果如图 2.27 所示。

来看一下其实现过程：




Note

当 n=0，则 2n=0，表示没有选中任何元素；

当 n=1，则 2n=2，表示选择了第 2 个 li；

当 n=2，则 2n＝4，表示选择了第 4 个 li。

依此类推。

如果是 2n，这样与使用 even 命名 class 定义样式所起到的效果是一样的。

☑ :nth-child(2n-1)

【示例 7】:nth-child(2n-1)选择器是在:nth-child(2n)基础上演变来的，既然:nth-child(2n)表示选择偶数，那么在它的基础上减去 1 就变成奇数选择。

```
#wrap li:nth-child(2n-1) {font-weight:bold;}
```

等价于：

```
#wrap li:nth-child(odd) {font-weight:bold;}
```

来看看其实现过程：

当 n=0，则 2n-1=-1，表示没有选中任何元素；

当 n=1，则 2n-1=1，表示选择第 1 个 li；

当 n=2，则 2n-1=3，表示选择第 2 个 li。

依此类推。

其实这种奇数效果，还可以使用:nth-child(2n+1)和:nth-child(odd)来实现。

☑ :nth-child(n+5)

【示例 8】:nth-child(n+5)选择器是从第 5 个子元素开始选择。

```
li:nth-child(n+5) {font-weight:bold;}
```

其实现过程如下：

当 n=0，则 n+5=5，表示选中第 5 个 li；

当 n=1，则 n+5=6，表示选择第 6 个 li。

依此类推。

读者可以使用这种方法选择需要开始选择的元素位置，也就是说换了数字，起始位置就变了。

☑ :nth-child(-n+5)

【示例 9】:nth-child(-n+5)选择器刚好和:nth-child(n+5)选择器相反，它是选择第 5 个前面的子元素。

```
li:nth-child(-n+5) {font-weight:bold;}
```

其实现过程如下：

当 n=0，则-n+5=5，表示选择了第 5 个 li；

当 n=1，则-n+5=4，表示选择了第 4 个 li；

当 n=2，则-n+5=3 ，表示选择了第 3 个 li；

当 n=3，则-n+5=2，表示选择了第 2 个 li；

当 n=4，则-n+5=1，表示选择了第 1 个 li；





当 n=5，则-n+5=0，表示没有选择任何元素。

☑　:nth-child(5n+1)

:nth-child(5n+1)选择器是实现隔几选一的效果。

【示例 10】如果是隔三选一，则定义的样式如下。

```
li:nth-child(3n+1) {font-weight:bold;}
```

其实现过程如下：

当 n=0，则 3n+1=1，表示选择了第 1 个 li；

当 n=1，则 3n+1=4，表示选择了第 4 个 li；

当 n=2，则 3n+1=7，表示选择了第 7 个 li。

设计效果如图 2.28 所示。



图 2.27　设计偶数行列表项样式



图 2.28　设计隔三选一行列表项样式

4. :nth-last-child()

【示例 11】:nth-last-child()选择器与:nth-child()相似，但作用与:nth-child()不一样，:nth-last-child()是从最后一个元素开始计算，来选择特定元素。

```
li:nth-last-child(4) {font-weight:bold;}
```

上面代码表示选择倒数第 4 个列表项。

其中，:nth-last-child(1)和:last-child 所起作用是一样的，都表示选择最后一个元素。

另外，:nth-last-child()与:nth-child()用法相同，可以使用表达式来选择特定元素，下面来看几个特殊的表达式所起的作用。

:nth-last-child(2n)表示从元素后面计算，选择的是偶数个数，从而反过来说就是选择元素的奇数，与前面的:nth-child(2n+1)、:nth-child(2n-1)、:nth-child(odd)所起的作用是一样的。例如：



```
li:nth-last-child(2n) {font-weight:bold;}
li:nth-last-child(even) {font-weight:bold;}
```

等价于：

```
li:nth-child(2n+1) {font-weight:bold;}
li:nth-child(2n-1) {font-weight:bold;}
li:nth-child(odd) {font-weight:bold;}
```

:nth-last-child(2n-1)选择器刚好与上面的相反，从后面计算选择的是奇数，而从前面计算选择的就是偶数位了。例如：

```
li:nth-last-child(2n+1) {font-weight:bold;}
li:nth-last-child(2n-1) {font-weight:bold;}
li:nth-last-child(odd) {font-weight:bold;}
```

等价于：

```
li:nth-child(2n) {font-weight:bold;}
li:nth-child(even) {font-weight:bold;}
```

总之，:nth-last-child()和 nth-child()的使用方法是一样的，区别是:nth-child()从元素的第一个开始计算，而:nth-last-child()是从元素的最后一个开始计算，它们的计算方法都是一样的。

5. :nth-of-type()

:nth-of-type()类似于:nth-child()，不同的是，:nth-of-type()只计算选择器中指定的那个元素。这个选择器在定位元素中包含很多不同类型的子元素时是很有用处的。

【示例 12】在 div#wrap 中包含很多 p、li、img 等元素，但现在只需要选择 p 元素，并让它每隔一个 p 元素就有不同的样式，可以简单写成：

```
div#wrap p:nth-of-type(even) {font-weight:bold;}
```

其实，这种用法与:nth-child()是一样的，也可以使用:nth-child()的表达式来实现，唯一不同的是，:nth-of-type()指定了元素的类型。

6. :nth-last-of-type()

:nth-last-of-type()与:nth-last-child()用法相同，但它指定了子元素的类型，除此之外，语法形式和用法基本相同。

7. :first-of-type()和:last-of-type()

:first-of-type()和:last-of-type()这两个选择器类似于:first-child()和:last-child()，不同之处就是它们指定了元素的类型。



8. :only-child 和:only-of-type

:only-child 表示的是一个元素是它的父元素的唯一一个子元素。

【示例 13】在文档中设计如下 HTML 结构。



```
<div class="post">
    <p>第一段文本内容</p>
    <p>第二段文本内容</p>
</div>
<div class="post">
    <p>第三段文本内容</p>
</div>
```

如果需要在 div.post 只有一个 p 元素的时候，改变这个 p 的样式，那么就可以使用:only-child 选择器来实现。

```
.post p {font-weight:bold;}
.post p:only-child {background: red;}
```

当 div.post 只有一个子元素 p 时，那么它的背景色将会显示为红色。

:only-of-type 表示一个元素包含很多个子元素，而其中只有一个子元素是唯一的，那么使用这种选择方法就可以选择这个唯一的子元素。例如：

```
<div class="post">
    <div>子块一</div>
    <p>文本段</p>
    <div>子块二</div>
</div>
```

如果只想选择上面结构块中的 p 元素，就可以这样写：

```
.post p:only-of-type{background-color:red;}
```

9. :empty

:empty 是用来选择没有任何内容的元素，这里没有内容指的是一点内容都没有，包括空格。

【示例 14】这里有 3 个段落，其中一个段落什么都没有，完全是空的。

```
<div class="post">
    <p>第一段文本内容</p>
    <p>第二段文本内容</p>
</div>
<div class="post">
```



```
    <p></p>
</div>
```



Note

如果想设计这个 p 不显示，那就可这样来写：

```
.post p:empty {display: none;}
```



视 频 讲 解

2.4.3　否定伪类

:not()表示否定选择器，即排除或者过滤掉特定元素。

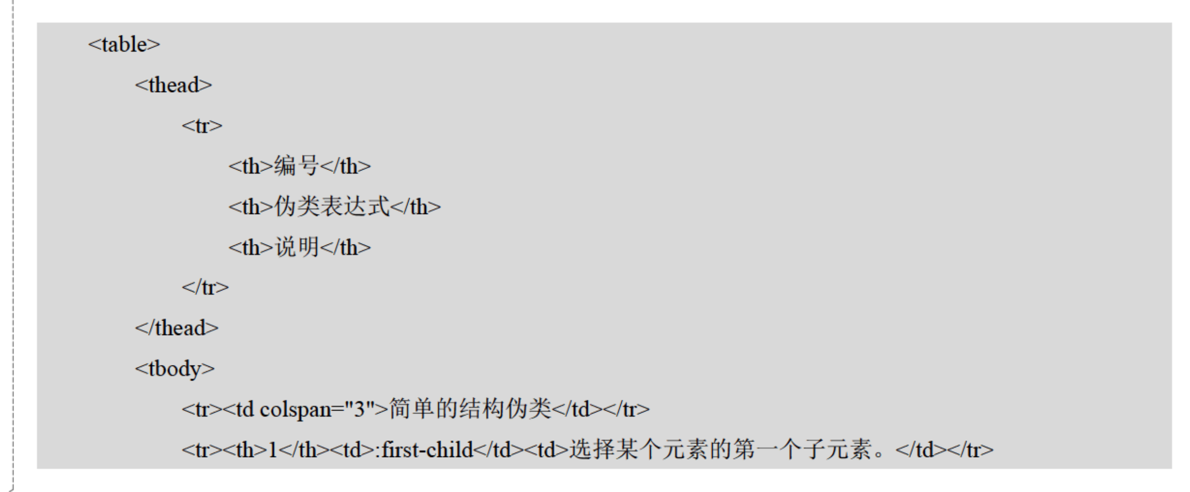
【示例】本示例是设计一个分层表格样式，主要借助否定伪类选择器和结构伪类选择器，配合 CSS 背景图像技术设计树形结构标志；借助伪类选择器设计鼠标经过时的动态背景效果；利用 CSS 边框和背景色设计标题行的立体显示效果。效果如图 2.29 所示。



图 2.29　设计表格样式

【操作步骤】

第 1 步，利用表格结构构建一个数据表。



```
<table>
    <thead>
        <tr>
            <th>编号</th>
            <th>伪类表达式</th>
            <th>说明</th>
        </tr>
    </thead>
    <tbody>
        <tr><td colspan="3">简单的结构伪类</td></tr>
        <tr><th>1</th><td>:first-child</td><td>选择某个元素的第一个子元素。</td></tr>
```



```
            <tr><th>2</th><td>:last-child</td><td>选择某个元素的最后一个子元素。</td></tr>
            <tr><th>3</th><td>:first-of-type</td><td>选择一个上级元素下的第一个同类子元素。</td></tr>
            <tr><th>4</th><td>:last-of-type</td><td>选择一个上级元素下的最后一个同类子元素。</td></tr>
            <tr><th>5</th><td>:only-child</td><td>选择的元素是它的父元素的唯一一个子元素。</td></tr>
            <tr><th>6</th><td>:only-of-type</td><td>选择一个元素是它的上级元素的唯一一个相同类型的子
元素。</td></tr>
            <tr><th class="end">7</th><td>:empty</td><td>选择的元素里面没有任何内容。</td></tr>
            <tr><td colspan="3">结构伪类函数</td></tr>
            <tr><th>8</th><td>:nth-child()</td><td>选择某个元素的一个或多个特定的子元素。</td></tr>
            <tr><th>9</th><td>:nth-last-child()</td><td>选择某个元素的一个或多个特定的子元素，从这个元
素的最后一个子元素开始算。</td></tr>
            <tr><th>10</th><td>:nth-of-type()</td><td>选择指定的元素。</td></tr>
            <tr><th class="end">11</th><td>:nth-last-of-type()</td><td>选择指定的元素，从元素的最后一个开
始计算。</td></tr>
        </tbody>
    </table>
```

第 2 步，使用<style>标签在当前文档中内建一个样式表，并初始化表格样式。

```
table { border-collapse: collapse; font-size: 75%; line-height: 1.4; border: solid 2px #ccc; width: 100%; }
th, td { padding: .3em .5em; cursor: pointer; }
th { font-weight: normal; text-align: left; padding-left: 15px; }
```

第 3 步，使用结构伪类选择器匹配合并单元格所在的行，定义合并单元格所在行加粗显示。

```
td:only-of-type {
    font-weight:bold;
    color:#444;}
```

第 4 步，使用否定伪类选择器选择主体区域非最后一个 th 元素。以背景方式在行前定义结构路径线。

```
tbody th:not(.end) {
    background: url(images/dots.gif) 15px 56% no-repeat;
    padding-left: 26px;}
```

第 5 步，使用类选择器选择主体区域非最后一个 th 元素。以背景方式在行前定义结构封闭路径线。

```
tbody th.end {
    background: url(images/dots2.gif) 15px 56% no-repeat;
    padding-left: 26px;}
```





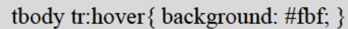
Note

第 6 步，使用 thead 元素把表头标题独立出来，方便 CSS 控制，避免定义过多的 class 属性。th 元素有两种显示形式，一种用来定义列标题，另一种用来定义行标题。下面是定义表格标题列样式。

```
thead th {
    background: #c6ceda;
    border-color: #fff #fff #888 #fff;
    border-style: solid;
    border-width: 1px 1px 2px 1px;
    padding-left: .5em;}
```

第 7 步，设计隔行换色的背景效果，这里主要应用了:nth-child(2n)选择器，同时使用:hover 动态伪类定义鼠标经过时的行背景色动画变化，以提示鼠标当前经过行效果。

```
tbody tr:nth-child(2n) {background-color: #fef;}
tbody tr:hover{ background: #fbf; }
```





视频讲解

2.4.4　状态伪类

CSS3 新增 3 个 UI 状态伪类选择器。简单说明如下。

1. :enabled

:enabled 伪类表示匹配指定范围内所有可用 UI 元素。在网页中，UI 元素一般是指包含在 form 元素内的表单元素。例如，在下面表单结构中，input:enabled 选择器将匹配文本框，但不匹配该表单中的按钮。

```
<form>
    <input type="text" /> disabled
    <input type="button"/>
</form>
```

2. :disabled

:disabled 伪类表示匹配指定范围内所有不可用 UI 元素。例如，在下面表单结构中，input:disabled 选择器将匹配按钮，但不匹配该表单中的文本框。

```
<form>
    <input type="text" />
    <input type="button" disabled="disabled" />
</form>
```

3. :checked

:checked 伪类表示匹配指定范围内所有可用 UI 元素。例如，在下面表单结构中，input:checked





选择器将匹配片段中的单选按钮，但不匹配该表单中的复选框。

```
<form>
    <input type="checkbox" />
    <input type="radio" checked="checked" />
</form>
```

【示例】本示例设计一个简单的登录表单，效果如图 2.30 所示。在实际应用中，当用户登录完毕，不妨通过脚本把文本框设置为不可用（disabled="disabled"）状态，这时可以通过:disabled 选择器让文本框显示为灰色，以告诉用户该文本框不可用了，这样就不用设计“不可用”样式类，并把该类添加到 HTML 结构中。

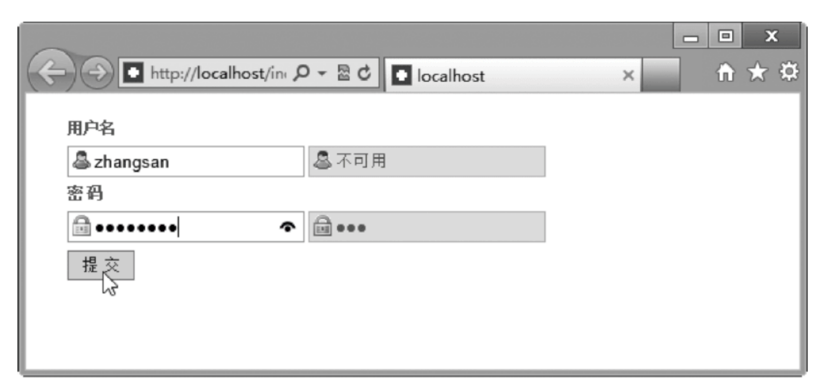


图 2.30　设计登录表单样式

【操作步骤】

第 1 步，新建一个文档，在文档中构建一个简单的登录表单结构。

```
<form action="#">
    <label for="username">用户名</label>
    <input type="text" name="username" id="username" />
    <input type="text" name="username1" disabled="disabled" value="不可用" />
    <label for="password">密码</label>
    <input type="password" name="password" id="password"   />
    <input type="password" name="password1" disabled="disabled" value="不可用"   />
    <input type="submit" value="提交" />
</form>
```

在这个表单结构中，使用 HTML 的 disabled 属性分别定义两个不可用的文本框对象。

第 2 步，内建一个内部样式表，使用属性选择器定义文本框和密码域的基本样式。

```
input[type="text"], input[type="password"] {
    border:1px solid #0f0;
```




Note

```
        width:160px;
        height:22px;
        padding-left:20px;
        margin:6px 0;
        line-height:20px;}
```

第 3 步，利用属性选择器分别为文本框和密码域定义内嵌标识图标。

```
input[type="text"] { background:url(images/name.gif) no-repeat 2px 2px; }
input[type="password"] { background:url(images/password.gif) no-repeat 2px 2px; }
```

第 4 步，使用状态伪类选择器定义不可用表单对象显示为灰色，以提示用户该表单对象不可用。

```
input[type="text"]:disabled {
    background:#ddd url(images/name1.gif) no-repeat 2px 2px;
    border:1px solid #bbb;}
input[type="password"]:disabled {
    background:#ddd url(images/password1.gif) no-repeat 2px 2px;
    border:1px solid #bbb;}
```


视频讲解

2.4.5 目标伪类

目标伪类选择器类型形式如 E:target，它表示选择匹配 E 的所有元素，且匹配元素被相关 URL 指向。该选择器是动态选择器，只有当存在 URL 指向该匹配元素时，样式效果才能够有效。

【示例】本示例设计了当单击页面中的锚点链接，跳转到指定标题位置时，该标题会自动高亮显示，以提醒用户当前跳转的位置，效果如图 2.31 所示。

```
<style type="text/css">
/* 设计导航条固定在窗口右上角位置显示 */
h1{ position:fixed; right:12px; top:24px;}
/* 让锚点链接堆叠显示*/
h1 a{ display:block;}
/* 设计锚点链接的目标高亮显示*/
h2:target { background:hsla(93,96%,62%,1.00); }
</style>

<h1><a href="#p1">图片 1</a><a href="#p2">图片 2</a><a href="#p3">图片 3</a><a href="#p4">图片 4</a></h1>
<h2 id="p1">图片 1</h2>
<p><img src="images/1.jpg" /></p>
<h2 id="p2">图片 2</h2>
```



```
<p><img src="images/2.jpg" /></p>
<h2 id="p3">图片 3</h2>
<p><img src="images/3.jpg" /></p>
<h2 id="p4">图片 4</h2>
<p><img src="images/4.jpg" /></p>
```

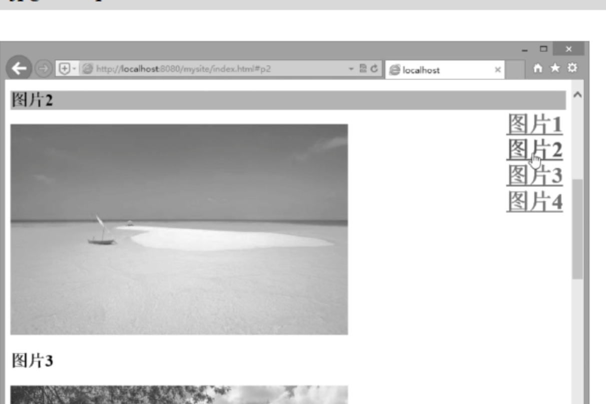


图 2.31　目标伪类样式应用效果

2.5　在线练习

本节为读者提供了多个在线练习，以便灵活使用 CSS，强化基本功训练。感兴趣的读者可以扫码练习。



在 线 练 习



第3章 使用 CSS 美化网页文本

网页的一个主要作用是传递信息，文本作为主要的信息载体，由以前的纯文本演变至丰富多彩的形式，主要归功于 CSS 样式。CSS 样式对文本的修饰不仅仅是字体类型、大小、粗细、倾斜等基本样式，还包括文本颜色、行高、下划线等各个方面的修饰。本章从基础的文字设置出发，详细讲解 CSS 设置各种文字效果的方法，然后进一步讲解段落排版的相关内容。

【学习要点】

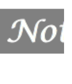
⏭ 定义字体样式。

⏭ 定义文本样式。

⏭ 能够灵活设计页面文本样式。









3.1　字体基本样式

网页字体样式包括字体类型、大小、颜色等基本效果，另外还包括一些特殊的样式，如字体粗细、下划线、斜体、大小写样式等。

3.1.1　定义字体类型

CSS 使用 font-family 属性来定义字体类型，用法如下。

```
font-family : name
font-family :ncursive | fantasy | monospace | serif | sans-serif
```

name 表示字体名称，可指定多种字体，多个字体将按优先顺序排列，以逗号隔开。如果字体名称包含空格，则应使用引号括起。第 2 种声明方式使用所列出的字体序列名称，如果使用 fantasy 序列，将提供默认字体序列。

【示例 1】新建一个网页，保存为 test.html，在<body>标签内输入一行段落文本。

```
<p>定义字体类型</p>
```

在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后输入下面样式，用来定义网页字体的类型。

```
body {/*  页面基本属性  */
    font-family:Arial, Helvetica, sans-serif;    /*  字体类型  */
}
p {/*  段落样式  */
    font:24px "隶书";                            /* 24 像素大小的隶书字体  */
}
```

font 是一个复合属性，能够设置多种字体属性，如字体大小、字体类型、行高、艺术字体等。用法如下。

```
font : font-style || font-variant || font-weight || font-size || line-height || font-family
font : caption | icon | menu | message-box | small-caption | status-bar
```

属性值之间以空格分隔。font 属性至少应设置字体大小和字体类型，且必须放在后面，否则无效。前面可以自由定义字体样式、字体粗细、大小写和行高，详细用法将在后面小节中分别介绍。

> **提示：**中文网页字体多定义为宋体类型，对于标题或特殊提示信息，如果需要特殊字体，则建议采用图像形式来间接实现。





Note



视频讲解

【示例 2】以列表的形式设置多种字体类型。在上面示例基础上，为段落文本设置 3 种字体类型，其中第一个为具体的字体类型，而后面两个为通用字体类型。

```
p {font-family:"Times New Roman", Times, serif}
```

注意，字体列表以逗号进行分隔，浏览器会根据这个字体列表来检索用户系统中的字库，按照从左到右的顺序进行选用。如果没有找到列表中对应的字体，则选用默认字体进行显示。

3.1.2　定义字体大小

CSS 使用 font-size 属性来定义字体大小，用法如下。

```
font-size : xx-small | x-small | small | medium | large | x-large | xx-large | larger | smaller | length
```

其中，xx-small（最小）、x-small（较小）、small（小）、medium（正常）、large（大）、x-large（较大）、xx-large（最大）表示绝对字体尺寸，这些特殊值将根据对象字体进行调整；larger（增大）和 smaller（减少）这对特殊值能够根据父对象中字体尺寸进行相对增大或者缩小处理，使用成比例的 em 单位进行计算；length 可以是百分数，或者浮点数字和单位标识符组成的长度值，但不可为负值。其百分比取值是基于父对象中字体的尺寸来计算，与 em 单位计算相同。

提示：CSS 提供了很多单位，它们可以被归为两大类：绝对单位和相对单位。

绝对单位所定义的字体大小是固定的，大小显示效果不会受外界因素影响。例如，in（inch，英寸）、cm（centimeter，厘米）、mm（millimeter，毫米）、pt（point，印刷的点数）、pc（pica，1pc=12pt）。此外，xx-small、x-small、small、medium、large、x-large、xx-large 这些关键字也是绝对单位。

相对单位所定义的字体大小一般是不固定的，会根据外界环境而不断发生变化，介绍如下。

☑ px（pixel，像素）：根据屏幕像素点的尺寸变化而变化。因此，不同分辨率的屏幕所显示的像素字体大小也是不同的，屏幕分辨率越大，相同像素字体就显得越小。

☑ em：相对于父辈字体的大小来定义字体大小。例如，如果父元素字体大小为 12 像素，而子元素的字体大小为 2em，则实际大小应该为 24 像素。

☑ ex：相对于父辈字体的 x 高度来定义字体大小，因此 ex 单位大小既取决于字体的大小，也取决于字体类型。在固定大小的情况下，实际的 x 高度将随字体类型不同而不同。

☑ %：以百分比的形式定义字体大小，它与 em 效果相同，相对于父辈字体的大小来定义字体大小。

☑ larger 和 smaller：这两个关键字将以父元素的字体大小为参考进行换算。

【示例 1】本示例演示了如何为网页文档定义字体大小。新建网页，保存为 test.html，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后输入下面样式，设置网页字体默认大小、正文字体大小，以及栏目中字体大小。

```
body {font-size:12px;}                    /* 以像素为单位设置字体大小 */
p {font-size:0.75em;}                     /* 以父辈字体大小为参考设置大小 */
```





```
div {font:9pt Arial, Helvetica, sans-serif;}      /*以点为单位设置字体大小*/
```

【示例 2】正确规划网页字体大小。网页设计师常用的字体大小单位包括像素和百分比，本示例围绕这两个单位进行讨论和练习。

☑　对于网页宽度固定或者栏目宽度固定的布局，使用像素是正确的。

☑　对于页面宽度不固定或者栏目宽度也不固定的页面，使用百分比或 em 是正确选择。

从用户易用性角度考虑，定义字体大小应该以 em（或%）为单位进行设置。主要考虑因素是：一方面，有利于客户端浏览器调整字体大小；另一方面，通过设置字体大小的单位为 em 或百分比，使字体能够适应版面宽度的变化。

【操作步骤】

第 1 步，新建一个网页，保存为 test1.html，在<body>标签内输入如下结构。

```
<div id="content">框架
    <div id="sub">子框架
        <p>段落文本</p>
    </div>
</div>
```

第 2 步，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表。然后定义样式，设计页面正文字体大小为 12 像素，使用 em 来设置，则代码如下。

```
body {/*  网页字体大小  */
    font-size:0.75em;          /*  约等于 12 像素  */
}
```

计算方法为：浏览器默认字体大小为 16 像素，用 16 像素乘以 0.75 即可得到 12 像素。同样的道理，预设 14 像素，则应该是 0.875em；预设 10 像素，则应该是 0.625em。

第 3 步，在复杂结构中如果反复选择 em 或百分比作为字体大小，可能会出现字体大小显示混乱的状况。如果修改上面代码中的样式，分别定义 body、div 和 p 元素的字体大小为 0.75em，由于 em 单位是以上级字体大小为参考进行显示，所以在浏览器中预览就会发现正文文字看不清楚，如图 3.1 所示。

```
body, div, p {font-size:0.75em;}
```

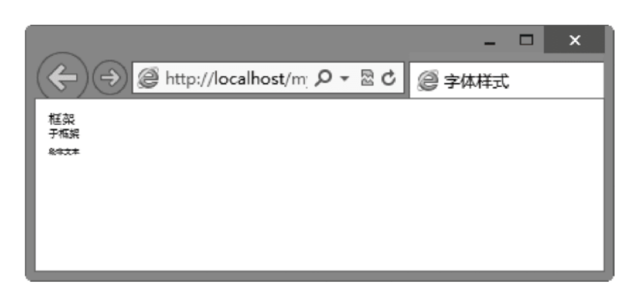


图 3.1　以 em 为单位所带来的隐患




提示：根据第 2 步的计算方法，body 字体大小应该为 12 像素，而<div id="content">内字体大小只为 9 像素，<div id="sub">内字体只为 7 像素，段落文本的字体大小只为 5 像素。所以，不要嵌套使用 em 单位定义字体大小。




视频讲解

3.1.3 定义字体颜色

CSS 使用 color 属性来定义字体颜色，用法如下。

```
color : color
```

参数 color 表示颜色值。

【示例】本示例演示了如何在文档中定义字体颜色。新建一个网页，保存为 test.html，在<head>标签内添加<style type="text/css">标签，定义内部样式表，然后输入下面样式，定义页面、段落文本、<div>标签、<span>标签包含的字体的颜色。

```
body { color:gray;}                    /*  使用颜色名  */
p { color:#666666;}                    /*  使用十六进制  */
div { color:rgb(120,120,120);}         /*  使用 RGB */
span { color:rgb(50%,50%,50%);}        /*  使用 RGB */
```

提示：CSS 3 新增以下 3 种颜色表示法。

☑ RGBA 颜色表示法

RGBA 颜色表示法是在 RGB 颜色的基础上增加了 Alpha 通道，这样就可以定义半透明的颜色。例如，color:rgba(255,0,0,5);声明可以定义半透明的红色。

☑ HSL 颜色表示法

HSL 颜色表示法是使用色相（H）、饱和度（S）和亮度（L）表示颜色的一种方法。例如，color:hsl(0, 100%,100%);表示红色。

☑ HSLA 颜色表示法

HSLA 颜色表示法是在 HSL 颜色的基础上增加了 Alpha 通道。例如，color:hsla(0, 100%,100%,5);表示半透明的红色。


视频讲解

3.1.4 定义字体粗细

CSS 使用 font-weight 属性来定义字体粗细，用法如下。

```
font-weight : normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900
```

font-weight 属性取值比较特殊，其中 normal 关键字表示默认值，即正常的字体，相当于取值为 400。bold 关键字表示粗体，相当于取值为 700 或者使用<b>标签定义的字体效果。bolder（较粗）和 lighter（较细）是相对于 normal 字体粗细而言的。



另外，也可以设置值为 100、200、300、400、500、600、700、800、900，它们分别表示字体的粗细，是对字体粗细的一种量化方式，值越大就表示越粗，相反就表示越细。

【示例】本示例演示如何定义网页对象的字体粗细样式。新建一个网页，保存为 test.html，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后输入下面样式，分别定义段落文本、一级标题、<div>标签包含字体的粗细效果，同时定义一个粗体样式类。

```
p { font-weight: normal }                /* 等于 400 */
h1 { font-weight: 700 }                  /* 等于 bold */
div{ font-weight: bolder }               /* 可能为 500 */
.bold {/* 粗体样式类 */
    font-weight:bold;                    /* 加粗显示 */
}
```

注意：设置字体粗细也可以称为定义字体的重量。对于中文网页设计来说，一般使用 bold（加粗）和 normal（普通）两个属性值即可。



视频讲

3.1.5　定义斜体字体

CSS 使用 font-style 属性来定义字体倾斜效果，用法如下。

```
font-style : normal | italic | oblique
```

其中，normal 表示默认值，即正常的字体；italic 表示斜体；oblique 表示倾斜的字体。italic 和 oblique 两个取值只在英文等西方文字中有效。

【示例】本示例演示了如何为网页定义斜体样式类。新建一个网页，保存为 test.html，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后输入下面样式，定义一个斜体样式类。

```
.italic {/* 斜体样式类 */
    font-style:italic;                   /* 斜体 */
}
```

在<body>标签中输入一行段落文本，并把斜体样式类应用到该段落文本中。

```
<p>古今之成大事业、大学问者，必经过三种之境界。<span class="italic">“昨夜西风凋碧树，独上高楼，望尽天涯路”</span>，此第一境也。<span class="italic">“衣带渐宽终不悔，为伊消得人憔悴”</span>，此第二境也。<span class="italic">“众里寻他千百度，蓦然回首，那人却在，灯火阑珊处。”</span>，此第三境也。此等语皆非大词人不能道。</p>
```



3.1.6　定义下划线



Note



视频讲解

CSS 使用 text-decoration 属性来定义字体下划线效果，用法如下。

```
text-decoration : none || underline || blink || overline || line-through
```

其中，none 表示默认值，即无装饰字体；underline 表示下划线效果；blink 表示闪烁效果；overline 表示上划线效果；line-through 表示删除线效果。

【操作步骤】

第 1 步，新建一个网页，保存为 test.html，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后输入下面样式，定义 3 个装饰字体样式类。

```
.underline {text-decoration:underline;}          /*下划线样式类 */
.overline {text-decoration:overline;}            /*上划线样式类 */
.line-through {text-decoration:line-through;}    /* 删除线样式类 */
```

第 2 步，在<body>标签中输入 3 行段落文本，并分别应用上面的装饰类样式。

```
<p class="underline">昨夜西风凋碧树，独上高楼，望尽天涯路</p>
<p class="overline">衣带渐宽终不悔，为伊消得人憔悴</p>
<p class="line-through">众里寻他千百度，蓦然回首，那人却在，灯火阑珊处。</p>
```

第 3 步，再定义一个样式，在该样式中，同时声明多个装饰值，定义的样式如下。

```
.line {text-decoration:line-through overlineunderline;}
```

第 4 步，在正文中输入一行段落文本，并把这个 line 样式类应用到该行文本中。

```
<p class="line">古今之成大事业、大学问者，必经过三种之境界。</p>
```

第 5 步，在浏览器中预览，可以看到最后一行文本显示多种修饰线效果，如图 3.2 所示。

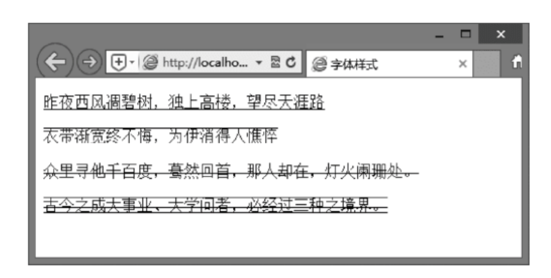


图 3.2　多种修饰线的应用效果



视频讲解

3.1.7　定义字体大小写

CSS 使用 font-variant 属性来定义字体大小写效果，该属性用法如下。





```
font-variant : normal | small-caps
```

其中，normal 表示默认值，即正常的字体；small-caps 表示小型的大写字母字体。

【示例 1】本示例演示了如何定义大写字体样式。新建一个网页，保存为 test.html，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后输入下面样式，定义一个类样式。

```
.small-caps {/* 小型大写字母样式类 */
    font-variant:small-caps;
}
```

然后在<body>标签中输入一行段落文本，并应用上面定义的类样式。

```
<p class="small-caps">font-variant </p>
```

注意，font-variant 仅支持以英文为代表的西文字体，中文字体没有大小写效果区分。如果设置了小型大写字体，但是该字体没有找到原始小型大写字体，则浏览器会模拟一个。例如，可通过使用一个常规字体，并将其小写字母替换为缩小过的大写字母。

提示：CSS 还定义了一个 text-transform 属性，该属性也能够定义字体大小写效果。不过该属性主要定义单词大小写样式，用法格式如下。

```
text-transform : none | capitalize | uppercase | lowercase
```

其中，none 表示默认值；capitalize 表示将每个单词的第一个字母转换成大写，其余无转换发生；uppercase 表示把所有字母都转换成大写；lowercase 表示把所有字母都转换成小写。

【示例 2】本示例借助 text-transform 属性定义首字母大写、全部大写和全部小写 3 个样式类。新建一个网页，保存为 test1.html，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后输入下面样式，定义 3 个类样式。

```
.capitalize {text-transform:capitalize;}/*首字母大小样式类 */
.uppercase {text-transform:uppercase;}/*大写样式类 */
.lowercase {text-transform:lowercase;}/* 小写样式类 */
```

在<body>标签中输入 3 行段落文本，并分别应用上面定义的类样式。

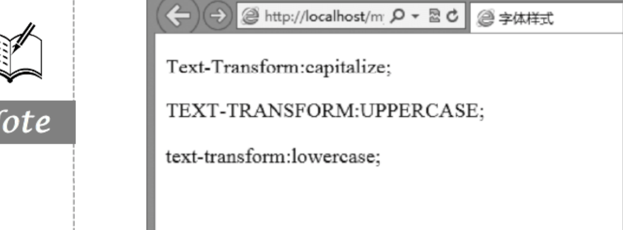
```
<p class="capitalize">text-transform:capitalize;</p>
<p class="uppercase">text-transform:uppercase;</p>
<p class="lowercase">text-transform:lowercase;</p>
```

分别在 IE 和 Firefox 浏览器中预览，则会发现：IE 认为只要是单词就把首字母转换为大写，如图 3.3 所示；而 Firefox 认为只有单词通过空格间隔之后，才能够成为独立意义上的单词，所以几个单词连在一起时就算作一个词，如图 3.4 所示。





Note



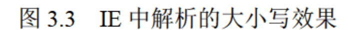


图 3.3　IE 中解析的大小写效果

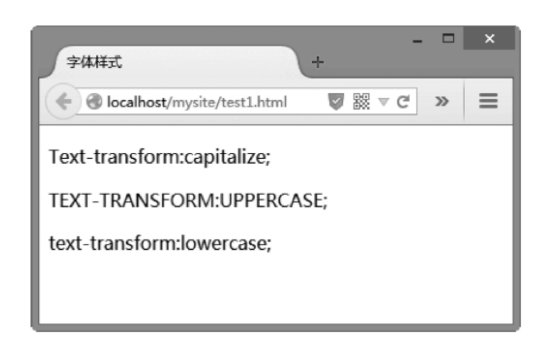


图 3.4　Firefox 中解析的大小写效果

3.2　文本基本样式

文本样式主要涉及多个字符的排版效果。CSS 在命名属性时，使用 font 前缀和 text 前缀来区分字体和文本属性。



视 频 讲 解

3.2.1　定义水平对齐

CSS 使用 text-align 属性来定义文本的水平对齐方式，用法如下。

```
text-align : left | right | center | justify
```

该属性取值包括 4 个：left 表示左对齐，默认值；right 表示右对齐；center 表示居中对齐；justify 表示两端对齐。CSS3 新增了 4 个属性：start | end | match-parent | justify-all，由于浏览器支持不是很好，读者可以先暂时了解。

【示例 1】 本示例定义文档中段落文本居中显示。新建一个网页，保存为 test.html，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后输入下面样式，定义居中对齐类样式。

```
.center { text-align:center;}/* 居中对齐类样式 */
```

在<body>标签中输入两行段落文本，并分别使用传统的 HTMLalign 属性和标准设计中 CSS 的 text-align 属性定义文本居中。

```
<p align="center">昨夜西风凋碧树，独上高楼，望尽天涯路</p><!-- 传统居中对齐方式 -->
<p class="center">衣带渐宽终不悔，为伊消得人憔悴</p>            <!-- 标准居中对齐方式 -->
<!-- 标准居中对齐方式 -->
<p class="center">众里寻他千百度，蓦然回首，那人却在，灯火阑珊处</p>
```

在浏览器中预览，可以看到使用传统方式和标准方式设计文本居中的效果是相同的。

【示例 2】 当 text-align 属性取值为 justify 时，可以结合 text-justify 属性实现更多对齐样式。本示



例定义第一段文本两端对齐，第二段文本保持默认对齐方式，效果如图 3.5 所示。

```
<style type="text/css">
#parag1 {/*强制所有文本行都两端对齐，包括最后一行，不够一行，会自动添加空格来实现两端对齐。该规则仅适合 IE 浏览器*/
    text-align:justify;
    text-justify:distribute-all-lines;
}
</style>

<p id="parag1">text-align:justify;</p>
<p>text-justify:distribute-all-lines;</p>
```





图 3.5　两端对齐和默认对齐显示

3.2.2　定义垂直对齐



CSS 使用 vertical-align 属性来定义文本垂直对齐方式，用法如下。

```
vertical-align : auto | baseline | sub | super | top | text-top | middle | bottom | text-bottom | length
```

取值简单说明如下。

- ☑ auto：表示将根据 layout-flow 属性的值对齐对象内容。
- ☑ baseline：默认值，表示将支持 valign 特性的对象内容与基线对齐。
- ☑ sub：表示垂直对齐文本的下标。
- ☑ super：表示垂直对齐文本的上标。
- ☑ top：表示将支持 valign 特性的对象的内容对象顶端对齐。
- ☑ text-top：表示将支持 valign 特性的对象的文本与对象顶端对齐。
- ☑ middle：表示将支持 valign 特性的对象的内容与对象中部对齐。
- ☑ bottom：表示将支持 valign 特性的对象的内容与对象底端对齐。
- ☑ text-bottom：表示将支持 valign 特性的对象的文本与对象底端对齐。
- ☑ length：表示由浮点数字和单位标识符组成的长度值或者百分数，可为负数，定义由基线算起的偏移量，基线对于数值来说为 0，对于百分数来说就是 0%。





Note

【示例 1】本示例将演示如何设置字体垂直对齐。新建一个网页，保存为 test1.html，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后输入下面样式，定义上标类样式。

```
.super {vertical-align:super;}
```

在<body>标签中输入一行段落文本，并应用该上标类样式。

```
<p>vertical-align 表示垂直<span class=" super ">对齐</span>属性</p>
```

在浏览器中预览，则显示效果如图 3.6 所示。



图 3.6　文本上标样式效果

【示例 2】本示例演示了不同垂直对齐方式的比较效果。

vertical-align 属性提供的值很多，但是 IE 浏览器与其他浏览器对于解析它们的效果却存在很大的分歧。一般情况下，不建议广泛使用这些属性值，实践中主要用到 vertical-align 属性的垂直居中样式，偶尔也会用到上标和下标效果。

新建一个网页，保存为 test2.html，在<body>标签内输入如下结构。

```
<p>valign:
<span class="baseline"><img src="images/box.gif" title="baseline" /></span>
<span class="sub"><img src="images/box.gif" title="sub" /></span>
<span class="super"><img src="images/box.gif" title="super" /></span>
<span class="top"><img src="images/box.gif" title="top" /></span>
<span class="text-top"><img src="images/box.gif" title="text-top" /></span>
<span class="middle"><img src="images/box.gif" title="middle" /></span>
<span class="bottom"><img src="images/box.gif" title="bottom" /></span>
<span class="text-bottom"><img src="images/box.gif" title="text-bottom" /></span>
</p>
```

在<head>标签内添加<style type="text/css">标签，定义内部样式表。然后定义如下类样式。

```
body {font-size:48px;}
.baseline {vertical-align:baseline;}
.sub {vertical-align:sub;}
```





```
.super {vertical-align:super;}
.top {vertical-align:top;}
.text-top {vertical-align:text-top;}
.middle {vertical-align:middle;}
.bottom {vertical-align:bottom;}
```

在浏览器中预览测试，则显示效果如图 3.7 所示。用户可以通过效果图直观地比较这些取值的效果。

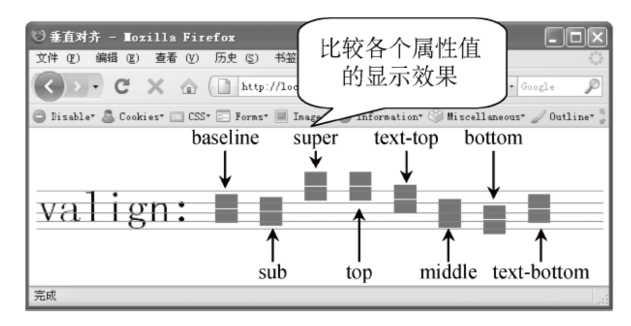


图 3.7　垂直对齐取值效果比较

3.2.3　定义字距和词距

CSS 使用 letter-spacing 属性定义字距，使用 word-spacing 属性定义词距。这两个属性的取值都是长度值，由浮点数字和单位标识符组成，默认值为 normal，表示默认间隔。

定义词距时，以空格为基准进行调节，如果多个单词连在一起，则被 word-spacing:视为一个单词；如果汉字被空格分隔，则分隔的多个汉字就被视为不同的单词，word-spacing 属性此时有效。

【示例】本示例演示了如何定义字距和词距样式。新建一个网页，保存为 test.html，在<head>标签内添加<style type="text/css">标签，定义内部样式表，然后输入下面样式，定义两个类样式。

```
.lspacing {letter-spacing:1em;}          /* 字距类样式 */
.wspacing {word-spacing:1em;}            /* 词距类样式 */
```

在<body>标签中输入两行段落文本，并应用上面两个类样式。

```
<p class="lspacing">letter spacing word spacing（字间距）</p>
<p class="wspacing">letter spacing word spacing（词间距）</p>
```

在浏览器中预览，则显示效果如图 3.8 所示。从图中可以直观地看到，所谓字距就是定义字母之间的间距，而词距就是定义西文单词的距离。





图 3.8　字距和词距演示效果比较




> 注意：字距和词距一般很少使用，使用时应慎重考虑用户的阅读体验和感受。对于中文用户来说，letter-spacing 属性有效，而 word-spacing 属性无效。



Note

视频讲解

3.2.4 定义行高

行高也称为行距，是段落文本行与文本行之间的距离。CSS 使用 line-height 属性定义行高，用法如下。

```
line-height : normal | length
```

其中，normal 表示默认值，一般为 1.2em；length 表示百分比数字，或者由浮点数字和单位标识符组成的长度值，允许为负值。

【示例 1】本示例演示了如何定义段落文本行高样式。新建一个网页，保存为 test.html，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后输入下面样式，定义两个行高类样式。

```
.p1 {/* 行高样式类 1 */
    line-height:1em;          /* 行高为一个字大小 */}
.p2 {/* 行高样式类 2 */
    line-height:2em;          /*行高为两个字大小 */}
```

在<body>标签中输入两行段落文本，并应用上面两个类样式。

```
<h1>人生三境界</h1>
<h2>出自王国维《人间词话》</h2>
<p class="p1">古今之成大事业、大学问者，必经过三种之境界："昨夜西风凋碧树。独上高楼，望尽天涯路。"此第一境也。"衣带渐宽终不悔，为伊消得人憔悴。"此第二境也。"众里寻他千百度，蓦然回首，那人却在，灯火阑珊处。"此第三境也。此等语皆非大词人不能道。然遽以此意解释诸词，恐为晏欧诸公所不许也。</p>
<p class="p2">笔者认为，凡人都可以从容地做到第二境界，但要想逾越它却不是那么简单。成功人士果敢坚忍，不屈不挠，造就了他们不同于凡人的成功。他们逾越的不仅仅是人生的境界，更是他们自我的极限。成功后回望来路的人，才会明白另解这三重境界的话：看山是山，看水是水；看山不是山，看水不是水；看山还是山，看水还是水。</p>
```

在浏览器中预览，则显示效果如图 3.9 所示。

【示例 2】本示例演示了不同取值下，行高样式的比较效果。行高取值单位一般使用 em 或百分比，很少使用像素，也不建议使用。

☑ 当 line-height 属性取值小于一个字大小时，就会发生上下行文本重叠的现象。在上面的示例基础上，修改定义的类样式。

```
.p1 { line-height:0.5em;}
.p2 { line-height:0em;}
```





在浏览器中预览，显示效果如图 3.10 所示，说明当取值小于字体大小时，多行文本会发生重叠现象。

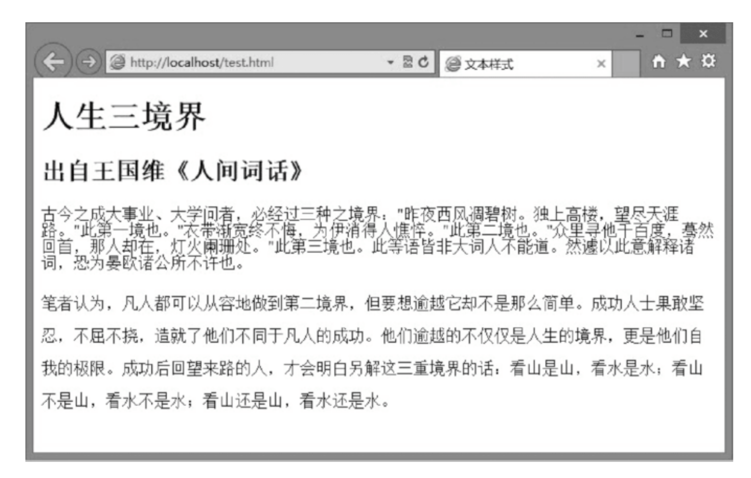


图 3.9　段落文本的行高演示效果



图 3.10　段落文本重叠演示效果

☑　一般行高的最佳设置范围为 1.2em~1.8em，当然对于特别大的字体或者特别小的字体，可以特殊处理。因此，用户可以遵循字体越大、行高越小的原则来定义段落行高。例如，如果段落字体大小为 12px，则行高设置为 1.8em 比较合适；如果段落字体大小为 14px，则行高设置为 1.5em~1.6em 比较合适；如果段落字体大小为 16px~18px，则行高设置为 1.2em 比较合适。一般浏览器默认行高为 1.2em 左右。例如，IE 默认为 19px，如果除以默认字体大小（16px），则约为 1.18em；而 Firefox 默认为 1.12em。

【示例 3】本示例演示了如何设置更灵活的行高样式。

用户可以给 line-height 属性设置一个数值，但是不设置单位。例如：

```
body { line-height:1.6;}
```

这时浏览器会把它作为 1.6em 或者 160%，也就是说页面行高实际为 19px。利用这种特殊的现象，可以解决多层嵌套结构中行高继承出现的问题。

新建一个网页，保存为 test2.html，在<head>标签内添加<style type="text/css">标签，定义一个内



部样式表，然后输入下面的样式，设置网页和段落文本的默认样式。




```
body {
    font-size:12px;
    line-height:1.6em;
    }
p { font-size:30px;}
```

在<body>标签中输入如下标题和段落文本。

```
<h1>《人间词话》节选</h1>
<h2>王国维</h2>
<p>古今之成大事业、大学问者，必经过三种之境界："昨夜西风凋碧树。独上高楼，望尽天涯路。"此第一境也。"衣带渐宽终不悔，为伊消得人憔悴。"此第二境也。"众里寻他千百度，蓦然回首，那人却在，灯火阑珊处。"此第三境也。此等语皆非大词人不能道。然遽以此意解释诸词，恐为晏欧诸公所不许也。</p>
```

上面示例定义 body 元素的行高为 1.6em。由于 line-height 具有继承性，因此网页中的段落文本的行高也继承 body 元素的行高。浏览器在继承该值时，并不是继承“1.6em”这个值，而是把它转换为精确值（即 19px）之后再继承，换句话说，p 元素的行高为 19px，但是 p 元素的字体大小为 30px，继承的行高小于字体大小，就会发生文本行重叠现象。如果在浏览器中预览，则演示效果如图 3.11 所示。



图 3.11　错误的行高继承效果

解决方法如下。

在定义 body 元素的行高时，不为其设置单位，即直接定义为 line-height:1.6，这样页面中其他元素所继承的值为 1.6，而不是 19px，则内部继承元素就会为继承的值 1.6 附加默认单位 em，最后页面中所有继承元素的行高都为 1.6em。

```
body {
    font-size:12px;
    line-height:1.6;
}
p { font-size:30px;}
```







3.2.5　定义缩进

CSS 使用 text-indent 属性定义首行缩进，用法如下。

```
text-indent : length
```

length 表示百分比数字，或者由浮点数字和单位标识符组成的长度值，允许为负值。建议在设置缩进单位时，以 em 为设置单位，它表示一个字距，这样比较精确确定首行缩进效果。

【示例 1】本示例演示了如何为段落文本定义首行缩进效果。新建一个网页，保存为 test.html，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后输入下面样式，定义段落文本首行缩进两个字距。

```
p {text-indent:2em;}                    /* 首行缩进两个字距 */
```

在<body>标签中输入如下标题和段落文本。

```
<h1>人生三境界</h1>
<h2>出自王国维《人间词话》</h2>
<p>古今之成大事业、大学问者，必经过三种之境界："昨夜西风凋碧树。独上高楼，望尽天涯路。"此第一境也。"衣带渐宽终不悔，为伊消得人憔悴。"此第二境也。"众里寻他千百度，蓦然回首，那人却在，灯火阑珊处。"此第三境也。此等语皆非大词人不能道。然遽以此意解释诸词，恐为晏欧诸公所不许也。</p>
<p>笔者认为，凡人都可以从容地做到第二境界，但要想逾越它却不是那么简单。成功人士果敢坚忍，不屈不挠，造就了他们不同于凡人的成功。他们逾越的不仅仅是人生的境界，更是他们自我的极限。成功后回望来路的人，才会明白另解这三重境界的话：看山是山，看水是水；看山不是山，看水不是水；看山还是山，看水还是水。</p>
```

在浏览器中预览，则可以看到文本缩进效果。

【示例 2】使用 text-indent 属性可以设计悬垂缩进效果。新建一个网页，保存为 test1.html，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后输入下面样式，定义段落文本首行缩进负的两个字符，并定义左侧内部补白为两个字符。

```
p {/* 悬垂缩进两个字距 */
    text-indent:-2em;                   /* 首行缩进*/
    padding-left:2em;                   /* 左侧补白 */
    }
```

text-indent 属性可以取负值，定义左侧补白，是为防止取负值缩进导致首行文本伸到段落的边界外边。

在<body>标签中输入如下标题和段落文本。

```
<h1>《人间词话》节选</h1>
<h2>王国维</h2>
```




```
<p>古今之成大事业、大学问者，必经过三种之境界："昨夜西风凋碧树。独上高楼，望尽天涯路。"此第一境也。"衣带渐宽终不悔，为伊消得人憔悴。"此第二境也。"众里寻他千百度，蓦然回首，那人却在，灯火阑珊处。"此第三境也。此等语皆非大词人不能道。然遽以此意解释诸词，恐为晏欧诸公所不许也。</p>
```

在浏览器中预览，则可以看到文本悬垂缩进效果，如图 3.12 所示。



图 3.12　悬垂缩进效果

3.3　CSS3 文本样式

CSS3 优化和增强了 CSS 2.1 的字体和文本属性，使网页文字更具表现力和感染力，丰富了网页文本的样式和版式。



视 频 讲 解

3.3.1　定义文本阴影

在 CSS3 中，可以使用 text-shadow 属性为文字添加阴影效果，用法如下。

```
text-shadow：none | <shadow> [ , <shadow> ]*
<shadow> = <length>{2,3} &&<color>?
```

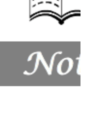
text-shadow 属性的初始值为无，适用于所有元素。取值简单说明如下。

- ☑ none：无阴影。
- ☑ <length>①：第 1 个长度值用来设置对象的阴影水平偏移值。可以为负值。
- ☑ <length>②：第 2 个长度值用来设置对象的阴影垂直偏移值。可以为负值，正值偏右或偏下，负值偏左或偏上。
- ☑ <length>③：如果提供了第 3 个长度值，则用来设置对象的阴影模糊值（即模糊半径）。不允许为负值。
- ☑ <color>：设置对象的阴影的颜色，该值可选。

【示例】 本示例为段落文本定义一个简单的阴影效果，演示效果如图 3.13 所示。

```
<style type="text/css">
p {
```






```
        text-align: center;
        font: bold 60px helvetica, arial, sans-serif;
        color: #999;
        text-shadow: 0.1em 0.1em #333;
        }
</style>
<p>文本阴影：text-shadow</p>
```



图 3.13　定义文本阴影

text-shadow: 0.1em 0.1em #333;声明了右下角文本阴影效果，如果把投影设置到左上角，则可以这样声明：

```
p {text-shadow: -0.1em -0.1em #333;}
```

演示效果如图 3.14 所示。

同理，如果设置阴影在文本的左下角，则可以设置如下样式。

```
p {text-shadow: -0.1em 0.1em #333;}
```

演示效果如图 3.15 所示。



图 3.14　定义左上角阴影



图 3.15　定义左下角阴影

也可以增加模糊效果的阴影：

```
p{text-shadow: 0.1em 0.1em 0.3em #333; }
```



效果如图 3.16 所示。

或者定义如下模糊阴影效果：



```
text-shadow: 0.1em 0.1em 0.2em black;
```

效果如图 3.17 所示。



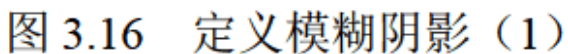


图 3.16　定义模糊阴影（1）



图 3.17　定义模糊阴影（2）

在阴影偏移之后，可以指定一个模糊半径。模糊半径是个长度值，指出模糊效果的范围。模糊效果的具体算法没有指定。在阴影效果的长度值之前或之后还可以选择指定一个颜色值。如果没有指定颜色，那么将使用 color 属性值来替代。

视频讲解

3.3.2　定义溢出文本

text-overflow 属性可以设置超长文本省略显示。基本语法如下。

```
text-overflow：clip | ellipsis
```

适用于块状元素，取值简单说明如下。

☑　clip：当内联内容溢出块容器时，将溢出部分裁切掉，为默认值。

☑　ellipsis：当内联内容溢出块容器时，将溢出部分替换为“...”。

提示： 在早期 W3C 文档（http://www.w3.org/TR/2003/CR-css3-text-20030514/#textoverflow-mode）中，text-overflow 被纳入规范，但是在最新修订的文档（http://www.w3.org/TR/css3-text/）中没有再包含 text-overflow 属性。

由于 W3C 规范放弃了对 text-overflow 属性的支持，所以，Mozilla 类型浏览器也放弃了对该属性的支持。不过，Mozilladevelopercenter 推荐使用-moz-binding 的 CSS 属性进行兼容。Firefox 支持 XUL（XUL 是一种 XML 的用户界面语言），这样就可以使用-moz-binding 属性来绑定 XUL 里的 ellipsis 属性了。





注意：text-overflow 属性仅是内容注解，即当文本溢出时是否显示省略标记，并不具备样式定义的特性。要实现溢出时产生省略号的效果，还应定义两个样式：强制文本在一行内显示（white-space:nowrap）和溢出内容为隐藏（overflow:hidden），只有这样才能实现溢出文本显示省略号的效果。

【示例】本示例设计了一个新闻列表有序显示效果，对于超出指定宽度的新闻项，则使用 text-overflow 属性省略并附加省略号，避免新闻换行或者撑开版块，演示效果如图 3.18 所示。



图 3.18　设计固定宽度的新闻栏目

示例代码如下。

```
<style type="text/css">
dl {/*定义新闻栏目外框，设置固定宽度*/
    width:300px;
    border:solid 1px #ccc;
}
dt {/*设计新闻栏目标题行样式*/
    padding:8px 8px;                                    /*增加文本周围空隙*/
    margin-bottom:12px;                                 /*调整底部间距*/
    background:#7FECAD url(images/green.gif) repeat-x;  /*设计标题栏背景图*/
 /*定义字体样式*/
    font-size:13px;font-weight:bold;color:#71790C;
    text-align:left;                                    /*恢复文本默认左对齐*/
    border-bottom:solid 1px #efefef;                    /*定义浅色边框线*/
}
dd {/*设计新闻列表项样式*/
    font-size:0.78em;
 /*固定每个列表项的大小*/
    height:1.5em;width:280px;
    /*为添加新闻项目符号腾出空间*/
```




Note

```
    padding:2px 2px 2px 18px;
    /*以背景方式添加项目符号*/
    background: url(images/icon.gif) no-repeat 6px 25%;
    margin:2px 0;
    /*为应用 text-overflow 做准备，禁止换行*/
    white-space: nowrap;
    /*为应用 text-overflow 做准备，禁止文本溢出显示*/
    overflow: hidden;
    -o-text-overflow: ellipsis;                      /* 兼容 Opera */
    text-overflow: ellipsis;                         /* 兼容 IE 和 Safari (WebKit) */
    -moz-binding: url('images/ellipsis.xml#ellipsis');   /* 兼容 Firefox */
    }
</style>

<dl>
    <dt>唐诗名句精选</dt>
    <dd>海内存知己，天涯若比邻。唐・王勃《送杜少府之任蜀州》</dd>
    <dd>不知细叶谁裁出，二月春风似剪刀。唐・贺知章《咏柳》</dd>
    <dd>欲穷千里目，更上一层楼。唐・王之涣《登鹳雀楼》</dd>
    <dd>野旷天低树，江清月近人。唐・孟浩然《宿建德江》</dd>
    <dd>大漠孤烟直，长河落日圆。唐・王维《使至塞上》</dd>
</dl>
```


视频讲解

3.3.3 定义文本换行

在 CSS3 中，使用 word-break 属性可以定义文本自动换行。基本语法如下。

```
word-break：normal | keep-all | break-all
```

取值简单说明如下。

☑ normal：为默认值，依照亚洲语言和非亚洲语言的文本规则，允许在字内换行。

☑ keep-all：对于中文、韩文、日文，不允许字断开。适合包含少量亚洲文本的非亚洲文本。

☑ break-all：与 normal 相同，允许非亚洲语言文本行的任意字内断开。该值适合包含一些非亚洲文本的亚洲文本，如使连续的英文字母间断行。

【补充】

word-break 原来是 IE 私有属性，在 CSS3 中被 Text 模块采用，得到 Chrome 和 Safari 等浏览器的支持，但不支持 keep-all 取值。

另外，IE 自定义了多个换行处理属性：line-break、word-break、word-wrap，CSS1 也定义了 white-space。这几个属性简单比较如下。



☑ line-break：指定如何（或是否）断行。除了 Firefox，其他浏览器都支持。取值说明如下所示。
- auto：使用默认的断行规则分解文本。
- loose：使用最松散的断行规则分解文本，一般用于短行的情况，如报纸。
- normal：使用最一般的断行规则分解文本。
- strict：使用最严格的断行规则分解文本。

☑ word-wrap：允许长单词或 URL 地址换行到下一行。所有浏览器都支持。取值说明如下所示。
- normal：只在允许的断字点换行（浏览器保持默认处理）。
- break-word：在长单词或 URL 地址内部进行换行。

☑ word-break：指定怎样在单词内断行。取值说明参考上面语法。

☑ white-space：设置如何处理元素中的空格。所有浏览器都支持。取值说明如下所示。
- normal：默认处理方式。
- pre：使用等宽字体显示预先格式化的文本，不合并文字间的空白距离，当文字超出边界时不换行。
- nowrap：强制在同一行内显示所有文本，合并文本间的多余空白。
- pre-wrap：使用等宽字体显示预先格式化的文本，不合并文字间的空白距离，当文字碰到边界时发生换行。
- pre-line：保持文本的换行，不保留文字间的空白距离，当文字碰到边界时发生换行。

【拓展】

在 IE 浏览器下，使用 word-wrap:break-word;声明可以确保所有文本正常显示。在 Firefox 浏览器下，中文不会出现任何问题，英文语句也不会出现问题。但是，长串英文会出现问题。为了解决长串英文会出现的问题，一般将 word-wrap:break-word;和 word-break:break-all;声明结合使用。但是，这种方法会导致普通的英文语句中的单词被断开显示（IE 下也是）的情况。现在的问题主要存在于长串英文和英文单词被断开的情况。

为了解决这个问题，可使用 word-wrap:break-word;overflow:hidden;，而不是 word-wrap:break-word; word-break:break-all;。word-wrap:break-word;overflow:auto;在 IE 下没有任何问题，但是在 Firefox 下，长串英文单词就会被遮住部分内容。

【示例】本示例将在页面中插入一个表格，由于标题行文字较多，标题行常被撑开，影响浏览体验。为了解决这个问题，借助 CSS 换行属性进行处理，代码如下。比较效果如图 3.19 所示。

```
<style type="text/css">
table {
    width: 100%;
    font-size: 14px;
    border-collapse: collapse;              /*定义细线表格*/
    border: 1px solid #cad9ea;              /*添加淡色细线边框*/
    table-layout: fixed;                    /*定义表格在浏览器端逐步解析呈现，避免破坏布局*/
```




Note

```
}
th {
    background-image: url(images/th_bg1.gif);      /*使用背景图模拟渐变背景*/
    background-repeat: repeat-x;                   /*定义背景图平铺方式*/
    height: 30px;
    vertical-align:middle;                         /*垂直居中显示*/
    border: 1px solid #cad9ea;                     /*添加淡色细线边框*/
    padding: 0 1em 0;
    overflow: hidden;                              /*超出范围隐藏显示，避免撑开单元格*/
    word-break: keep-all;                          /*禁止词断开显示*/
    white-space: nowrap;                           /*强迫在一行内显示*/
}
td {
    height: 20px;
    border: 1px solid #cad9ea;                     /*添加淡色细线边框*/
    padding: 6px 1em;                              /*增加单元格空隙，避免文本挤在一起*/
}
tr:nth-child(even) { background-color: #f5fafe; }
.w4 { width: 4em; }
</style>

<table>
  <tr>
    <th class="w4">与文本换行相关的属性</th>
    <th>使用说明</th>
  </tr>
  <tr>
    <td>line-break</td>
    <td>......</td>
  </tr>
  <tr>
    <td>word-wrap</td>
    <td>......</td>
  </tr>
  <tr>
    <td>word-break</td>
    <td>......</td>
  </tr>
```







```
    <tr>
        <td>white-space</td>
        <td>......</td>
    </tr>
</table>
```



与文本换行相关的属性	使用说明
line-break	用于指定如何（或是否）断行。除了Firefox，其它浏览器都支持。取值包括： auto，使用缺省的断行规则分解文本； loose，使用最松散的断行规则分解文本，一般用于短行的情况，如报纸； normal，使用最一般的断行规则分解文本； strict，使用最严格的断行原则分解文本。
word-wrap	允许长单词或URL地址换行到下一行。所有浏览器都支持。取值包括： normal，只在允许的断字点换行（浏览器保持默认处理）； break-word，在长单词或 URL 地址内部进行换行。
word-break	定义文本自动换行。Chrome和Safari浏览器支持不够友好。取值包括： normal：为默认值，允许在字内换行; keep-all,对于中文、韩文、日文，不允许字断开； break-all，与normal相同，允许非亚洲语言文本行的任意字内断开。
white-space	设置如何处理元素中的空格。所有浏览器都支持。取值包括： normal，默认处理方式； pre，显示预先格式化的文本，当文字超出边界时不换行； nowrap，强制在同一行内显示所有文本，合并文本间的多余空白，直到文本结束或者遭遇br对象； pre-wrap，显示预先格式化的文本，不合并文字间的空白距离，当文字碰到边界时发生换行； pre-line，保持文本的换行，不保留文字间的空白距离，当文字碰到边界时发生换行。

处理前



与文本换行	使用说明
line-break	用于指定如何（或是否）断行。除了Firefox，其它浏览器都支持。取值包括： auto，使用缺省的断行规则分解文本； loose，使用最松散的断行规则分解文本，一般用于短行的情况，如报纸； normal，使用最一般的断行规则分解文本； strict，使用最严格的断行原则分解文本。
word-wrap	允许长单词或URL地址换行到下一行。所有浏览器都支持。取值包括： normal，只在允许的断字点换行（浏览器保持默认处理）； break-word，在长单词或 URL 地址内部进行换行。
word-break	定义文本自动换行。Chrome和Safari浏览器支持不够友好。取值包括： normal：为默认值，允许在字内换行; keep-all,对于中文、韩文、日文，不允许字断开； break-all，与normal相同，允许非亚洲语言文本行的任意字内断开。
white-space	设置如何处理元素中的空格。所有浏览器都支持。取值包括： normal，默认处理方式； pre，显示预先格式化的文本，当文字超出边界时不换行； nowrap，强制在同一行内显示所有文本，合并文本间的多余空白，直到文本结束或者遭遇br对象； pre-wrap，显示预先格式化的文本，不合并文字间的空白距离，当文字碰到边界时发生换行； pre-line，保持文本的换行，不保留文字间的空白距离，当文字碰到边界时发生换行。

处理后

图 3.19　禁止表格标题文本换行显示

3.3.4　动态添加文本



content 属性属于内容生成和替换模块，可以为匹配的元素动态生成内容，这样就能够满足在 CSS 样式设计中临时添加非结构性的样式服务标签，或者添加补充说明性内容等。

content 属性的简明语法如下。

```
content: normal | string | attr() | url() | counter() | none;
```

取值简单说明如下。

☑ normal：默认值。表现与 none 值相同。

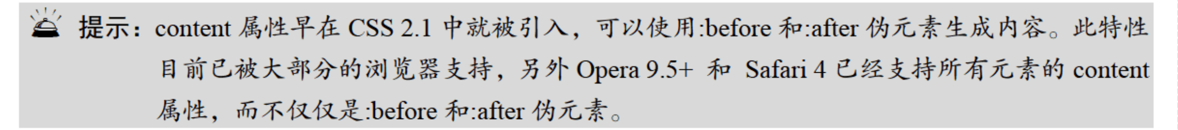
☑ string：插入文本内容。

☑ attr()：插入元素的属性值。

☑ url()：插入一个外部资源，如图像、音频、视频或浏览器支持的其他任何资源。

☑ counter()：计数器，用于插入排序标识。

☑ none：无任何内容。



提示： content 属性早在 CSS 2.1 中就被引入，可以使用:before 和:after 伪元素生成内容。此特性目前已被大部分的浏览器支持，另外 Opera 9.5+ 和 Safari 4 已经支持所有元素的 content 属性，而不仅仅是:before 和:after 伪元素。





在 CSS 3 Generated Content 工作草案中，content 属性添加了更多的特征，例如，插入以及移除文档内容的能力，可以创建脚注、结语和段落注释。但是目前还没有浏览器支持 content 的扩展功能。

【示例】本示例使用 content 属性，配合 CSS 计数器设计多层嵌套有序列表序号设计，代码如下。效果如图 3.20 所示。

```
<style type="text/css">
ol { list-style:none;}                                        /*清除默认的序号*/
li:before {color:#f00; font-family:Times New Roman;}          /*设计层级目录序号的字体样式*/
li{counter-increment:a 1;}                                    /*设计递增函数 a，递增起始值为 1*/
li:before{content:counter(a)". ";}                            /*把递增值添加到列表项前面*/
li li{counter-increment:b 1;}                                 /*设计递增函数 b，递增起始值为 1*/
li li:before{content:counter(a)"."counter(b)". ";}            /*把递增值添加到二级列表项前面*/
li li li{counter-increment:c 1;}                              /*设计递增函数 c，递增起始值为 1*/
li li li:before{content:counter(a)"."counter(b)"."counter(c)". ";}  /*把递增值添加到三级列表项前面*/
</style>
<ol>
    <li>一级列表项目 1
        <ol>
            <li>二级列表项目 1</li>
            <li>二级列表项目 2
                <ol>
                    <li>三级列表项目 1</li>
                    <li>三级列表项目 2</li>
                </ol>
            </li>
        </ol>
    </li>
    <li>一级列表项目 2</li>
</ol>
```

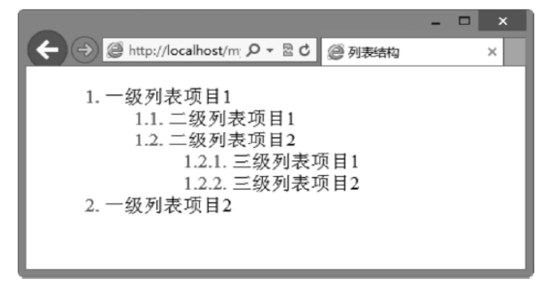


图 3.20　使用 CSS 技巧设计多级层级目录序号







3.3.5　自定义字体类型

CSS3 允许用户通过@font-face 规则，加载网络字体文件，实现自定义字体类型的功能。@font-face 规则在 CSS3 规范中属于字体模块。

@font-face 规则的语法格式如下。

```
@font-face { <font-description> }
```

@font-face 规则的选择符是固定的，用来引用网络字体文件。<font-description>是一个属性名值对，格式类似如下样式。

```
descriptor: value;
descriptor: value;
descriptor: value;
descriptor: value;
[...]
descriptor: value;
```

属性及其取值说明如下。

- ☑ font-family：设置文本的字体名称。
- ☑ font-style：设置文本样式。
- ☑ font-variant：设置文本是否大小写。
- ☑ font-weight：设置文本的粗细。
- ☑ font-stretch：设置文本是否横向拉伸变形。
- ☑ font-size：设置文本字体大小。
- ☑ src：设置自定义字体的相对路径或者绝对路径。注意，该属性只能在@font-face 规则里使用。

提示： 事实上，IE 5 已经开始支持 font-face 属性，但是只支持微软自有的.eot（Embedded OpenType）字体格式，而其他浏览器直到现在都不支持这一字体格式。不过，从 Safari 3.1 开始，用户可以设置.ttf（TrueType）和.otf（OpenType）两种字体作为自定义字体。考虑到浏览器的兼容性，在使用时建议同时定义.eot 和.ttf，以便能够兼容所有主流浏览器。

【示例】 这是一个简单的示例，演示如何使用@font-face 规则在页面中使用网络字体。示例代码如下，演示效果如图 3.21 所示。

```
<style type="text/css">
/*  引入外部字体文件  */
@font-face {
```




Note

```
        /* 选择默认的字体类型 */
        font-family: "lexograph";
        /* 兼容 IE */
        src: url(http://randsco.com//fonts/lexograph.eot);
        /* 兼容非 IE */
        src: local("Lexographer"), url(http://randsco.com/fonts/lexograph.ttf) format("truetype");
    }
    h1 {
        /* 设置引入字体文件中的 lexograph 字体类型 */
        font-family: lexograph, verdana, sans-serif;
        font-size:4em;
        }
    </style>

    <h1>http://www.baidu.com/</h1>
```



图 3.21　设置为 lexograph 字体类型的文字

> **提示：** 嵌入外部字体需要考虑用户带宽问题，因为一个中文字体文件小的有几个 MB，大的有十几个 MB，这么大的字体文件其下载过程会出现延迟，同时服务器也不能忍受如此频繁的申请下载。如果只是想让标题使用特殊字体，最好设计成图片。

3.4　案 例 实 战

下面结合案例讲解网页字体样式和文本版式设计，同时介绍各种网页设计的技巧。



视 频 讲 解

3.4.1　设计 Logo 样式

本案例将利用 CSS3 自定义字体模拟百度公司 Logo 字体样式，设计效果如图 3.22 所示。百度 Logo 的字体样式：“百度”二字是在“综艺体”的基础上稍加修改而成，英文字体是 Handel Gothic BT。







图 3.22　模拟百度 Logo 效果

【操作步骤】

第 1 步，新建 HTML 文档，保存为 index.html。构建简单的网页结构，其中<p>标签中包含两个<span>标签和一个<img>标签，预览效果如图 3.23 所示。

```
<p>
    <span class="g1">Bai</span>
    <img src="images/baidu.jpg" border="0">
    <span class="g2">百度</span>
</p>
```



图 3.23　构建百度 Logo 示例的页面结构

第 2 步，规划整个页面的基本显示属性，包括字体颜色、字体基本类型、网页字体大小等。由于本页面中的字体颜色是一致的，所以在<p>标签中定义了网页的字体颜色，并让文本居中显示。

```
p {
    color: #eb0005;
    text-align: center
}
```



第 3 步，使用@font-face 引入外部字体文件。



```
@font-face {
    font-family: "bai";                               /*  选择默认的字体类型  */
    src: url(fonts/Handel.eot);                       /*  兼容 IE */
    src: local("bai"), url(fonts/Handel.ttf) format("truetype");   /*  兼容非 IE */
}
@font-face {
    font-family: "du";
    src: url(fonts/方正新综艺简体.eot);
    src: local("du"), url(fonts/方正新综艺简体.ttf) format("truetype");
}
```

第 4 步，分别设置两个<span>标签的样式。由于在本案例中，既有中文又有西文，而中文和西文在显示上差别较大，所以分别进行设置，本案例中对第一个<span>，也就是英文“Bai”的样式设置如下。

```
.g1{
    font-size:60px;                               /*  字体大小*/
    font-family:MS Ui Gothic,Arial,sans-serif;    /*  字体类型*/
    letter-spacing:1px;                           /*  字间距  */
    font-weight:bold;                             /*  字体粗细*/
}
```

第 5 步，设置第 2 个<span>，也就是中文“百度”。具体类样式如下。

```
.g2{
    font-size:50px;
    font-family:MS Ui Gothic,Arial,sans-serif;
    letter-spacing:1px;
    font-weight:900;                              /*  字体粗细为 900 */
}
```

第 6 步，使用相对定位 position: relative;设置中间的图标向下偏移 8px，让图像与文字垂直居中对齐显示。

```
p img { position: relative; top: 8px;}
```



3.4.2 设计标题样式

本案例以设计标题为例，介绍 CSS 在控制文字时的各种技法，效果如图 3.24 所示。







图 3.24　标题样式的演示效果

【操作步骤】

第 1 步，新建 HTML 文档，保存为 index.html。

第 2 步，构建网页结构。在本案例中使用了<h1>标签，并加入了<img>标签，用于插入图片进行装饰。

```
<h1><img  src="images/tang.png" >《唐诗三百首》</h1>
```

第 3 步，定义网页基本属性。定义背景为 bg.jpg 的图片，上下左右的边界均为 20px，字体大小为 16px，字体为黑体。此时的显示效果如图 3.25 所示。

```
body {/*  页面基本属性*/
    margin:20px;                                        /*  边界*/
    background:url(images/bg.jpg);                      /*  背景图片*/
    font-size:14px;                                     /*  网页字体大小*/
    font-family:"宋体", Arial, Helvetica, sans-serif;    /*  网页字体默认类型*/
}
```



图 3.25　定义网页基本属性

第 4 步，定义标题样式。居中显示，字体颜色为#086916。




```
h1 {
    text-align:center;                    /* 居中对齐*/
    color:#086916;                        /* 字体颜色*/
}
```

此时<h1>标签加入了简单的 CSS 样式，包括对齐方式和字体颜色，效果如图 3.26 所示。

第 5 步，在文字下面添加一条 2px 宽的灰色边框，以增强效果，并在文字的下方增加补白，适当调整标题底边界。

```
h1{
    color:#086916;
    text-align:center;
    padding-bottom:24px;              /* 定义底边界为 24px*/
    border-bottom:2px solid #cecaca;  /* 宽为 2px 的灰色下边框*/
}
```

此时的效果如图 3.27 所示。由于<h1>是一个块级元素，所以它的边界不仅仅到文字，而是与页面的水平宽度灵活地保持一致。需要特别指出的是，这种创建边框的方法（border-bottom:2px solid #cecaca）是一个由 3 个部分组成的语句：宽度、样式、颜色。读者可以试着改变它们的值，看看会产生什么不同的效果。



图 3.26　对标题进行简单的 CSS 设置



图 3.27　添加灰色下边框

第 6 步，定义<img>标签的图片样式。为了使图片位置下移，对图片进行了相对定位 position: relative，并向下移动 24px。

```
img {
    position: relative;
    height:80px;
    bottom: -24px;
}
```







3.4.3　设计正文样式

本案例通过设置正文样式进一步介绍 CSS 文字和段落的排版方法，效果如图 3.28 所示。



图 3.28　正文样式效果

【操作步骤】

第 1 步，构建网页结构。本案例中<h1>标签与上例相同，同时增加了 3 个<p>标签，添加段落文本。此时显示效果仅仅是简单的文字和标题，如图 3.29 所示。

```
<h1><img  src="images/tang.png" >唐诗欣赏</h1>
<p>清明时节雨纷纷，路上行人欲断魂。借问酒家何处有？牧童遥指杏花村。——杜牧《清明》</p>
<p>空山新雨后，天气晚来秋。明月松间照，清泉石上流。竹喧归浣女，莲动下渔舟。随意春芳歇，王孙自可留。——王维《山居秋暝》</p>
<p>慈母手中线，游子身上衣。临行密密缝，意恐迟迟归；谁言寸草心，报得三春晖。——孟郊《游子吟》</p>
```



图 3.29　未加入 CSS 样式的网页基本结构

第 2 步，定义网页基本属性和标题属性，与上例基本相同。

```
body {
margin:30px;                                  /*边界*/
background:url(images/bg.jpg);                /* 背景图片*/
```




Note

```
        font-size:14px;                                  /* 网页字体大小*/
        font-family:"宋体", Arial, Helvetica, sans-serif;   /* 网页字体默认类型*/
    }
    h1 {
        font-family: 隶书; font-size: 50px; color: #086916; text-align: center;
        padding-bottom: 24px; border-bottom: 2px solid #cecaca;
    }
    img { position: relative; height:60px; bottom: -14px;}
```

第 3 步，设置正文样式，即<p>标签中的段落内容。此时<p>标签加入了 CSS 样式，包括字体大小、字体颜色和行间距等，但是并没有设置字体类型，所以<p>将会继承其父级属性，显示为宋体。

```
p{
    line-height:1.6em;                  /*行间距*/
    font-size:13px;                     /*字体大小*/
    color:#000;                         /*字体颜色*/
    text-indent:2em;                    /*定义首行缩进两个字符*/
    margin:0;                           /*四周补白为 0 */
}
```



视频讲解

3.4.4 规划网页字体大小

利用 em 和%作为网页字体大小的单位，可以设计出一套科学的网页字体大小方案。本案例设计网页字体大小配置方案如下。

☑ 网站标题字体大小为 16 像素。
☑ 栏目标题字体大小为 14 像素。
☑ 导航菜单字体大小为 13 像素。
☑ 正文字体大小为 12 像素。
☑ 版权、注释信息字体大小为 11 像素。

【操作步骤】

第 1 步，新建文档，在<body>标签内输入下面代码，定义网页的 HTML 框架。

```
<div id="wrap">
    <div id="header">
        <h1>网站标题（<span style="font-size:16px;">网站标题-16px</span>）</h1>
    </div>
    <ul id="nav">
        <li>菜单（<span style="font-size:13px;">菜单-13px</span>）</li>
    </ul>
    <div id="main">
```





```
        <h2>栏目标题（<span style="font-size:14px;">栏目标题-14px</span>）</h2>
            <p>网页正文（<span style="font-size:12px;">网页正文-12px</span>）</p>
        </div>
        <div id="footer">
            <p>版权信息（<span style="font-size:11px;">版权信息-11px</span>）</p>
        </div>
    </div>
```

第 2 步，新建内部样式表，定义网页字体大小，(12px/16px)×1em = 0.75em，也就是说初始化网页字体大小为 0.75em（相当于 12px），代码如下。

```
body {
    font-size:0.75em;
}
```

第 3 步，以 body 元素的字体大小为参考，来定义其他栏目或版块的字体大小。

☑ 网站标题的字体大小：(16px/12px)×1em = 1.333em，也就是说网站标题的字体大小是 body 字体大小的 16/12 倍，即等于 1.33em。

为什么不是(16px/12px)×0.75em = 1em？因为 body 的字体大小被定义为 0.75em。

根据 CSS 继承规则，子元素的字体大小都是以父元素的字体大小为 1em 作为参考来计算的，也就是说，如果网站标题定义为 1em，而 body 字体大小为 0.75em，则网站标题也应该为 0.75em，即等于 12px，而不是 16px。

☑ 栏目标题的字体大小：(14px/12px)×1em = 1.167em，也就是说，栏目标题的字体大小是 body 字体大小的 14/12 倍，即等于 1.167em。

☑ 导航菜单的字体大小：(13px/12px)×1em = 1.08em，也就是说，栏目标题的字体大小是 body 字体大小的 13/12 倍，即等于 0.812em。

☑ 正文的字体大小：(12px/12px)×1em = 1em，也就是说，正文的字体大小是 body 字体大小的 1 倍，即等于 1em。

☑ 版权、注释信息的字体大小：(11px/12px)×1em = 0.917em，也就是说，版权、注释信息的字体大小是 body 字体大小的 11/12 倍，即等于 0.917em。

第 4 步，针对上面的 HTML 结构，定义的 CSS 样式如下，其中正文字体直接继承 body 元素的字体大小，因此不需要重复定义，演示效果如图 3.30 所示。

```
<style type="text/css">
body {font-size:0.75em;}
#header h1 {font-size:1.333em;}
#main h2 {font-size:1.167.em;}
#nav li{font-size:1.08em;}
 #footer p {font-size:0.917em;}
</style>
```




Note

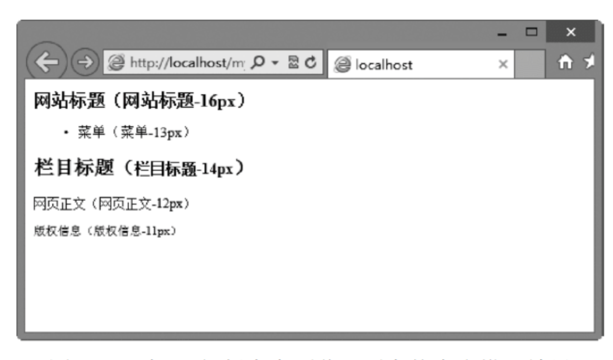


图 3.30　在 IE 新版本中预览网页字体大小搭配效果

上面的字体大小配置方案，适合用在嵌套层次比较浅的字体大小继承中，且要注意相互的干扰性。例如，如果创建一个样式 ol {font-size:60%;}，那么在列表嵌套中就会出现严重问题，内部的<ol>标签所包含的字体会实际显示为 36%（60%×60%）。所以，在使用 em 为单位定义字体大小时，要考虑网页结构的层次问题，原则上不要嵌套使用以 em 为单位定义字体大小超过两层，否则会为网页字体大小的统筹设计带来很多麻烦。


视 频 讲 解

3.4.5　设计居中显示

CSS 的 text-align 属性仅能够作用于文本等行内对象，而无法对块元素进行对齐操作。

【示例 1】本示例的代码在标准浏览器中是无法居中显示的，如图 3.31 所示。不过在 div 元素内包含文本可以居中显示，这是因为 text-align 属性拥有继承特性。

```
<style type="text/css">
body {text-align:center;}
div {
    border:solid 1px red;
    width:60%;}
</style>
<div><img src="images/1.png" /></div>
```



图 3.31　网页默认对齐方式



【示例 2】在现代标准浏览器中，可以通过定义 margin 属性实现居中显示，即定义其左右边距都为自动，则标准浏览器就会自动把块状元素置于居中的位置，代码如下。

```
<style type="text/css">
body { text-align: center; }
div {
    margin-left: auto;
    margin-right: auto;
    border: solid 1px red;
    width: 60%;
}
</style>
<div><img src="images/1.png" /></div>
```

【提示】

当网页嵌套层次比较深时，所设置的样式相互影响，由于对齐属性具有继承性，如果在 body 元素中声明居中对齐（text-align:center;），则网页内所有文本都会居中对齐。为了避免类似问题，必须在内部声明向左对齐进行纠正。

【示例 3】对于下面这个框架结构：

```
<div id="wrap">
    <h2>标题文本</h2>
    <div id="main"></div>
    <div id="footer"></div>
</div>
```

如果希望网页居中显示，则可以定义如下样式来兼容不同类型的浏览器。

```
body {
    text-align:center;          /*  定义网页在 IE 下对齐  */
    }
#wrap {
    margin:0 auto;              /*  定义网页在标准浏览器中对齐  */
}
```

虽然上面的方法实现了网页在不同类型浏览器中的对齐效果，但是文本也跟着居中对齐了，为了防止此问题的发生，可以在#wrap 选择器中补加一条规则，代码如下。

```
#wrap {
    margin:0 auto;
    text-align:left;
}
```



这样所有问题都解决了。如果希望网页内某个元素内文本居中对齐，则只需要单独定义一个样式即可。例如，再补加一个样式声明标题文本居中对齐，代码如下。



Note

```
#wrap h2 {
    text-align:center;
}
```



视频讲解

3.4.6 设计对象垂直对齐

【示例 1】各主流浏览器对 vertical-align 的支持并不统一。输入下面的代码，会发现在 IE 或 Firefox 等不同类型浏览器中所显示的效果都没有对齐底部，如图 3.32 所示。

```
<style type="text/css">
div {
    vertical-align: bottom;
    width: 12em;
    height: 6em;
    border: solid 1px red;}
</style>
<div>文本垂直对齐</div>
```



图 3.32　无效的垂直对齐底部

原来，vertical-align 仅能够作用于单元格或图像显示。因此，如果要在上面的样式内增加 display:table-cell;声明，则在标准浏览器中能够正确显示，如图 3.33 所示。

```
<style type="text/css">
div {
    vertical-align: bottom;
    display: table-cell;
    width: 12em;
    height: 6em;
    border: solid 1px red;
}
<div>文本垂直对齐</div>
```







图 3.33　垂直对齐底部显示

如果在表格单元格标签内定义 vertical-align 属性，则不同类型的浏览器都能够很好地支持。

【示例 2】对于下面的垂直对齐样式，IE 和 Firefox 等浏览器的解析效果是相同的。

```
<style type="text/css">
.cell {
    vertical-align: bottom;
    height: 60px;
}
</style>
<table width="200" border="1">
    <tr>
        <td class="cell">文本垂直对齐</td>
    </tr>
</table>
```

但是在其他元素内，IE 怪异模式就不能够很好地支持 vertical-align 属性了，即使声明了 display: table-cell;也是如此。为此只能另辟蹊径，下面介绍一下单行文本垂直居中对齐设计技巧。

【示例 3】单行文本垂直居中对齐是经常需要解决的问题，可以使用下面的方法巧妙地解决。

```
<style type="text/css">
div {
    line-height: 6em;
    width: 12em;
    height: 6em;
    border: solid 1px red;}
</style>
<div>文本垂直居中对齐</div>
```

通过定义单行文本的高度和行高相同，就能够间接地实现文本垂直居中显示的问题了，如图 3.34 所示。当然对于多行文本来说，这种方法无效。







图 3.34　单行文本垂直居中显示



3.4.7　隐藏和截取网页文字

在页面制作的过程中，经常会考虑如何控制页面中某个区域的文字内容的量，使其不会因为内容过多而撑开容器，甚至导致页面的错位。

【示例】在一个宽度为 300px、高度为 54px 的段落<p>标签中有一大段文字，由于文字过多导致无法正常显示在段落<p>标签内，代码如下，效果如图 3.35 所示。

```
<style type="text/css">
p {
    width: 300px;
    height: 54px;
    background-color: #EEEEEE;
}
</style>
<p>迎涛神，此说出自东汉<span>《曹娥碑》</span>。曹娥是东汉上虞人，父亲溺于江中，数日不见尸体，当时孝女曹娥年仅十四岁，昼夜沿江号哭。过了十七天，在五月五日也投江，五日后抱出父尸。</p>
```

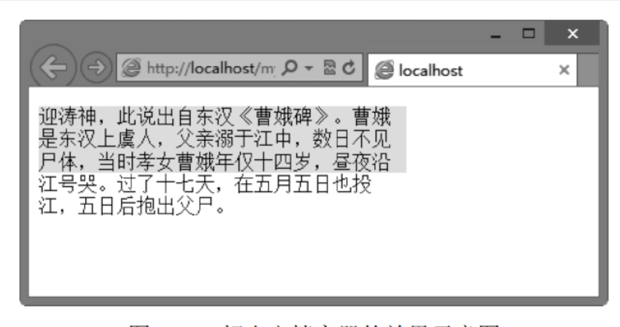


图 3.35　超出文档容器的效果示意图

但根据 CSS 样式所定义的，只需要段落<p>标签的高度是 54px，多余的应该是不需要的。给段落<p>标签的样式加上 overflow 属性，让多余的部分“消失”，代码如下。





```
p {
    width:300px;
    height:54px;
    overflow:hidden; /* 隐藏超出段落<p>标签容器的内容 */
    background-color:#EEEEEE;
}
```

添加 overflow:hidden;让超出段落<p>标签容器的部分在页面中“消失”，效果如图 3.36 所示。



图 3.36　添加 overflow:hidden;后段落<p>标签的表现形式效果图

文字隐藏的功能并不仅仅表现在能解决页面错位的问题上，还可以实现以图代替文字显示在页面中。所谓以图代替文字，其实就是隐藏文字，然后以背景图的方式显示文字。这种方式很常用，因为在设计页面时经常会有比较美观的被处理过的文字，如图 3.37 所示。



图 3.37　页面中经过处理的文字

经过处理的文字效果是用图片在页面中表现，但又不希望 HTML 结构中是使用图片<img>标签插入，而是使用<h1>标题标签表明该图片是一个标题，而且是全文中权重值最高的标题。那么 HTML 结构就会如此编码：

```
<h1>乐淘正品鞋城</h1>
```

在前面已经讨论过，如果要将文字隐藏，必须将容器的宽高固定，并且设置隐藏，现在要添加一张图片做背景，设置 CSS 样式代码如下，效果如图 3.38 所示。

```
h1 {
    width:250px;
    height:80px;
```





```
    overflow:hidden;
    background:url(images/logo.jpg) no-repeat 0 0;
}
```



图 3.38　并未“消失”的文字

既然已经设置 overflow:hidden;为什么文字还在呢？其实忘了一件很重要的事情，那就是只有当容器中的内容超过容器的宽高后才会隐藏。

在分析首行缩进时，曾学习了如何利用 text-indent 属性隐藏文字。现在就是 text-indent 发挥其作用的时候了。修改 CSS 样式代码，利用 text-indent 属性将文字往旁边“推”，远远地“抛”出容器之外。

```
h1 {
    width:250px;
    height:80px;
    overflow:hidden;
    text-indent:-9999px; /* 利用 text-indent 属性将文字“推”到容器之外 */
    background:url(images/logo.jpg) no-repeat 0 0;
}
```

在浏览器中预览效果，如图 3.39 所示，文字“消失”了，以图代替文字的方法生效。



图 3.39　文字“消失”（1）

文字既然可以左右移动很大的数值导致其超出容器的宽度而隐藏，那么如果将行高的值设置很大并超出容器的高度，不是也可以隐藏文字吗？





```
h1 {
    width:250px;
    height:80px;
    overflow:hidden;
    line-height:9999px; /* 将行高的值设置很大，超出容器之外，使其不可见 */
    background:url(images/logo.jpg) no-repeat 0 0;
}
```

如图 3.40 所示，文字因为行高的关系被“推”到了容器之外，并隐藏了。

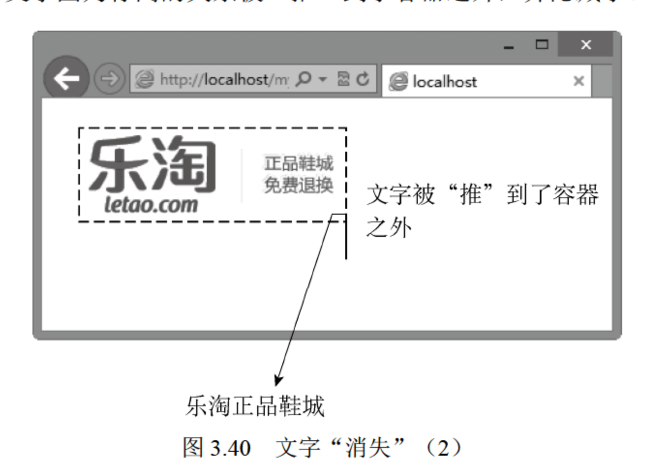


图 3.40　文字“消失”（2）

CSS 样式对于隐藏文字的处理不仅仅只有将元素“推”出容器之外这一个方法，还可利用 CSS 样式本身所具备隐藏特性的属性。

☑　visibility:hidden;可设置元素不可见，但会占据页面中其原本所应该占有的空间位置。

☑　display:none;可设置元素不可见，不占据页面中任何空间位置。

这两种方式的唯一区别就是，是否还会在原有的位置上保留其不可见后的元素空间，相同之处就是标签元素内的内容不可见。

使用这两种方式都需要在<h1>标题标签内多添加一个标签，这里添加一个<span>标签，代码如下。

```
<h1><span>乐淘正品鞋城</span></h1>
```

那么样式中首先需要设置<h1>标签的宽高以及背景图片的属性；其次，要对<h1>标题标签内的<span>标签中的元素设置不可见，代码如下。

```
h1 {
    width:250px;
    height:80px;
    background:url(images/logo.jpg) no-repeat 0 0;
}
h1 span {
```




```
        visibility:hidden; /*  设置<span>标签内的文字不可见，但在页面中占据其原本所占据的空间  */
    }
```



最终虽然文字“消失”了，但是在其原有的位置上还是保留着消失之前的空间位置。了解了关于使用 visibility:hidden;方法隐藏文字之后，再看一下使用 display:none;隐藏文字后的效果，代码如下。

```
h1 {
    width:250px;
    height:80px;
    background:url(images/logo.jpg) no-repeat 0 0;
}
h1 span {
    display:none; /*  设置<span>标签内的文字不可见，并且不会在页面中占据其空间  */
}
```

修改 CSS 样式中对<h1>标题标签所包含的<span>标签的样式定义方式，把原有的 visibility: hidden;隐藏文字方法改成 display:none;的方法来隐藏文字。利用 Firebug 也并无发现隐藏后的文字还保留着其原有的物理空间。

隐藏截取文字的方式虽然有多种，但并不是任何时候都是可行的，还应根据实际的情况去选用。只有掌握了如何使用这些方法，才能够设计出适合当前页面的效果。



视频讲解

3.4.8 设计文章版式

本案例将展示一个简单的中文版式：段落文本缩进，标题居中显示，正文之前设计一个题引，题引为左右缩进的段落文本显示效果，正文以首字下沉效果显示。演示效果如图 3.41 所示。



图 3.41　报刊式中文格式效果

【操作步骤】

第 1 步，设计网页结构。本示例的 HTML 文档结构依然采用禅意花园的结构，截取第一部分的结构和内容，并把英文全部译为中文。



```
<div id="intro">
    <div id="pageHeader">
        <h1><span>CSSZenGarden</span></h1>
        <h2><span><acronym title="cascading style sheets">CSS</acronym>设计之美</span></h2>
    </div>
    <div id="quickSummary">
    <p class="p1"><span>展示以<acronym
title="cascading style sheets">CSS</acronym>技术为基础，并提供超强的视觉冲击力。只要选择列表中任意
一个样式表，就可以将它加载到本页面中，并呈现不同的设计效果。</span></p>
<p class="p2"><span>下载<a title="这个页面的 HTML 源代码不能够被改动。"
href="http://www.csszengarden.com/zengarden-sample.html">HTML 文档</a>和<a
title="这个页面的 CSS 样式表文件，你可以更改它。"
href="http://www.csszengarden.com/zengarden-sample.css">CSS 文件</a>。</span></p>
    </div>
    <div id="preamble">
        <h3><span>启蒙之路</span></h3>
        <p class="p1"><span>不同浏览器随意定义标签，导致无法相互兼容的<acronym
title="document object model">DOM</acronym>结构，或者提供缺乏标准支持的<acronym
title="cascading style sheets">CSS</acronym>等陋习随处可见，如今当使用这些不兼容的标签和样式时，设
计之路会一路坎坷。</span></p>
<p class="p2"><span>现在，我们必须清除以前为了兼容不同浏览器而使用的一些过时的小技巧。感谢
<acronym
title="world wide web consortium">W3C</acronym>、<acronym
title="web standards project">WASP</acronym>等标准组织，以及浏览器厂家和开发师们的不懈努力，我们
终于能够进入 Web 设计的标准时代。</span></p>
        <p class="p3"><span>CSS Zen
            Garden（样式表禅意花园）邀请你发挥自己的想象力，构思一个专业级的网页。让我们用慧
眼来审视，充满理想和激情去学习 CSS 这个不朽的技术，最终使自己能够达到技术和艺术合而为一的最高境
界。</span></p>
    </div>
</div>
```

第 2 步，定义网页基本属性。定义背景色为白色，字体为黑色。多数浏览器默认网页就是这个样式，但是考虑到部分浏览器会以灰色背景显示，显式声明这些基本属性会更加安全。设置字体大小为 14px，字体为宋体。具体代码如下。

```
body {/* 页面基本属性*/
    background:#fff;                    /* 背景色*/
    color:#000;                         /* 前景色*/
```




Note

```
        font-size:0.875em;                              /* 网页字体大小*/
        font-family:"新宋体", Arial, Helvetica, sans-serif;   /* 网页字体默认类型*/
    }
```

第 3 步，定义标题居中显示，适当调整标题底边距，统一为一个字距。间距设计的一般规律：字距小于行距，行距小于段距，段距小于块距。检查的方法可以尝试将网站的背景图案和线条全部去掉，看是否还能保持想要的区块感。

```
    h1, h2, h3 {/* 标题样式*/
        text-align:center;                    /* 居中对齐*/
        margin-bottom:1em;                    /* 定义底边界*/
        }
```

第 4 步，为二级标题定义一个下划线，并调暗字体颜色，目的是使一级标题、二级标题和三级标题在同一个中轴线显示时产生一个变化，避免单调。由于三级标题字数少（4 个汉字），可以通过适当调节字距来设计一种平衡感，避免因为字数太少而使标题看起来很单调。

```
    h2 {/* 个性化二级标题样式*/
        color:#999;                           /* 字体颜色*/
        text-decoration:underline;            /* 下划线*/
        }
    h3 {/* 个性化三级标题样式*/
        letter-spacing:0.4em;                 /* 字距*/
        font-size:1.4em;                      /* 字体大小*/
        }
```

第 5 步，定义段落文本的样式。统一清除段落间距为 0，定义行高为 1.8 倍字体大小。

```
    p {/* 统一段落文本样式*/
        margin:0;                             /* 清除段距*/
        line-height:1.8em;                    /* 定义行高*/
        }
```

第 6 步，定义第一文本块中的第一段文本字体为深灰色，定义第一文本块中的第二段文本右对齐，定义第一文本块中的第一段和第二段文本首行缩进两个字距，同时定义第二文本块的第一段、第二段和第三段文本首行缩进两个字距。

```
    #quickSummary .p1 {/* 第一文本块的第一段样式*/
        color:#444;                           /* 字体颜色*/
        }
    #quickSummary .p2 {/* 第一文本块的第二段样式*/
```





```
        text-align:right;                          /* 右对齐*/
    }
#quickSummary .p1, .p2, .p3 {/* 除了首字下沉段以外的段样式*/
        text-indent:2em;                           /* 首行缩进*/
    }
```

第 7 步，为第一个文本块定义左右缩进样式，设计引题的效果。

```
#quickSummary {/* 第一文本块样式*/
        margin-left:4em;                           /* 左缩进*/
        margin-right:4em;                          /* 右缩进*/
    }
```

第 8 步，定义首字下沉效果。CSS 提供了一个首字下沉的属性：first-letter，这是一个伪对象（关于伪、伪类和伪对象，将在超链接设计章节中进行详细讲解）。但是 first-letter 属性所设计的首字下沉效果存在很多问题，所以还需要进一步设计。例如，设置段落首字浮动显示（关于浮动请参阅 CSS 布局章节），同时定义字体大小很大，以实现下沉效果。为了使首字下沉效果更明显，这里设计首字加粗、反白显示。

```
.first:first-letter {/* 首字下沉样式类*/
        font-size:50px;                            /* 字体大小*/
        float:left;                                /* 向左浮动显示*/
        margin-right:6px;                          /* 增加右侧边距*/
        padding:2px;                               /* 增加首字四周的补白*/
        font-weight:bold;                          /* 加粗字体*/
        line-height:1em;                           /* 定义行距为一个字体大小，避免行高影响段落版式*/
        background:#000;                           /* 背景色*/
        color:#fff;                                /* 前景色*/
    }
```

注意，由于 IE 早期版本浏览器存在 Bug，无法通过:first-letter 选择器来定义首字下沉效果，故这里重新定义了一个首字下沉的样式类（first），然后手动把这个样式类加入 HTML 文档结构对应的段落中。

```
<p class="p1 first"><span>不同浏览器随意定义标签，导致无法相互兼容的<acronym
title="document object model">DOM</acronym>结构，或者提供缺乏标准支持的<acronym
title="cascading style sheets">CSS</acronym>等陋习随处可见，如今当使用这些不兼容的标签和样式时，设
计之路会很坎坷。</span></p>
```





Note

提示：在阅读信息时，段落文本的呈现效果多以块状存在。如果说单个字是点，一行文本为线，那么段落文本就成面了，而面以方形呈现的效率最高，网站的视觉设计大部分其实都是在拼方块。在页面版式设计中，建议坚持如下设计原则。

☑ 方块感越强，越能给用户方向感。

☑ 方块越少，越容易阅读。

☑ 方块之间以空白的形式进行分隔，从而组合为一个更大的方块。

其他样式以及整个案例效果请参阅本节实例源代码。

3.5 在线练习

使用 CSS 设计各种文本效果，以及各种网页版式和文本特效。感兴趣的读者可以扫码练习。



在线练习1



在线练习2



使用 CSS 美化图像

在网页中，可以使用<img>标签插入图像，也可以以背景的形式显示图像。但不管使用哪种方式，都需要使用 CSS 对其进行美化。本章将分别介绍插入图像和背景图像的样式，帮助用户设计美观、大气的页面效果。

【学习要点】

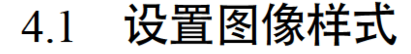
⏭ 设置插入图像的样式。

⏭ 设计背景图像样式。

⏭ 灵活使用图像美化网页效果。







4.1 设置图像样式

HTML5 为<img>标签定义了多个可选属性，不再推荐使用 HTML4 中部分属性，如 align（水平对齐方式）、border（边框粗细）、hspace（左右空白）、vspace（上下空白），对于这些属性，HTML5 建议使用 CSS 属性代替。



视频讲解

4.1.1 定义图像大小

<img>标签包含 width 和 height 属性，使用它们可以控制图像的大小。不过 CSS 提供了更符合标准的 width 和 height 属性，使用这两个属性可以更灵活地设计图像大小。

【示例 1】这是一个简单地使用 CSS 控制图像大小的案例。启动 Dreamweaver，新建网页，保存为 test1.html，在<body>标签内输入以下代码。

```
<img class="w200" src="images/1.jpg" />
<img class="w200" src="images/2.jpg" />
<img class="w200" src="images/3.jpg" />
<img src="images/4.jpg" />
```

在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后输入下面样式，以类样式的方式控制网页中图片的显示大小。

```
.w200 { /* 定义控制图像高度的类样式 */
    height:200px;
}
```

显示效果如图 4.1 所示，可以看到使用 CSS 更方便控制图片大小，提升了网页设计的灵活性。当图像大小取值为百分比时，浏览器将根据图像包含框的宽和高进行计算。

【示例 2】在这个示例中，统一定义图像缩小 50%，然后分别放在网页中和一个固定大小的盒子中，则显示效果截然不同，比较效果如图 4.2 所示。

```
<style type="text/css">
div { /* 定义固定大小的包含框 */
    height:200px;                     /* 固定高度 */
    width:50%;                        /* 设计弹性宽度 */
    border:solid 1px red;             /* 定义一个边框 */
    }
img { /* 定义图像大小 */
    width:50%;                        /* 百分比宽度 */
    height:50%;                       /* 百分比高度 */
```





```
    }
</style>
<div> <img src="images/4.png" /> </div>
<img src="images/4.png" />
```



图 4.1　固定缩放图像

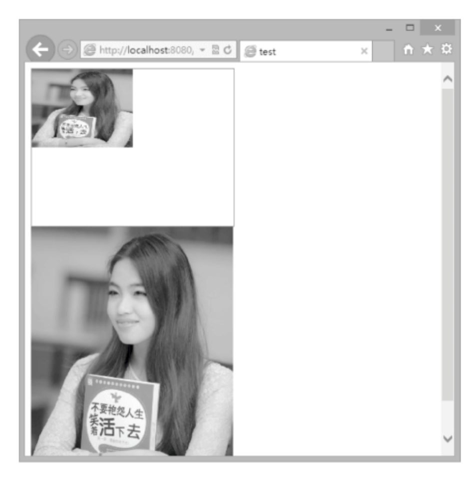


图 4.2　百分比缩放图像

提示： 当仅为图像定义宽度或高度时，则浏览器能够自动调整纵横比，使宽和高能够协调缩放，避免图像变形。但是一旦同时为图像定义宽和高，就要注意宽高比，否则会失真。

4.1.2　定义图像边框



图像在默认状态下是不会显示边框的，但在为图像定义超链接时会自动显示 2～3 像素宽的蓝色粗边框。使用 border 属性可以清除这个边框，代码如下。

```
<a href="#"><img src="images/login.gif" alt="登录" border="0" /></a>
```

不推荐上述用法，建议使用 CSS 的 border 属性定义。CSS 的 border 属性不仅可以为图像定义边框，而且提供了丰富的边框样式，支持定义边框的粗细、颜色和样式。

【示例 1】 针对上面的清除图像边框效果，使用 CSS 定义的代码如下。

```
img { /* 清除图像边框 */
    border:none;
}
```

使用 CSS 为<img>标签定义无边框显示，这样就不再需要为每个图像定义 0 边框的属性。下面分别讲解图像边框的样式、颜色和宽度的详细设置方法。

1．边框样式

CSS 为元素边框定义了众多样式，边框样式可以使用 border-style 属性来定义。边框样式包括两





Note

种：虚线框和实线框。

☑ 虚线框包括 dotted（点）和 dashed（虚线）。

【示例 2】在本示例中，分别定义两个不同的点线和虚线类样式，然后分别应用到两幅图像上，效果如图 4.3 所示，通过比较可以看到点线和虚线的细微差异。

```
<style type="text/css">
img {width:250px; margin:12px;}  /* 固定图像显示大小 */
.dotted { /* 点线框样式类 */
    border-style:dotted;}
.dashed { /* 虚线框样式类 */
    border-style:dashed;
        }
</style>
<img class="dotted" src="images/1.png" alt="点线边框" />
<img class="dashed" src="images/1.png" alt="虚线边框" />
```



图 4.3　IE 浏览器中的点线和虚线比较效果

☑ 实线框包括实线框（solid）、双线框（double）、立体凹槽（groove）、立体凸槽（ridge）、立体凹边（inset）和立体凸边（outset）。其中实线框（solid）是应用最广的一种边框样式。

提示：双线框由两条单线和中间的空隙组成，三者宽度之和等于边框的宽度。但是，双线框的值分配也会存在一些矛盾，无法做到平均分配。如果边框宽度为 3px，则两条单线与其间空隙分别为 1px；如果边框宽度为 4px，则外侧单线为 2px，内侧和中间空隙分别为 1px；如果边框宽度为 5px，则两条单线宽度为 2px，中间空隙为 1px。其他取值依此类推。

2. 边框颜色和宽度

使用 CSS 的 border-color 属性可以定义边框的颜色；使用 border-width 属性可以定义边框的宽度。当元素的边框样式为 none 时，所定义的边框颜色和边框宽度都会同时无效。在默认状态下，元素的边框样式为 none，而元素的边框宽度默认为 2～3px。





【示例 3】在本示例中快速定义图像各边的边框，显示效果如图 4.4 所示。

```
<style type="text/css">
img {/* 图像的边框样式 */
    width:100px;                          /* 宽度 */
    border:solid red 150px;               /* 统一定义各边样式：实线框、红色、150 像素宽度 */
    border-color:red blue green yellow;   /* 顶边红色、右边蓝色、底边绿色、左边黄色 */
}
</style>
<img src="images/1.png" />
```

【示例 4】可以配合使用不同的复合属性自定义各边样式，例如，本示例分别用 border-style、border-color 和 border-width 属性自定义图像各边边框样式，演示效果如图 4.5 所示。

```
<style type="text/css">
img {/* 图像的边框样式 */
    width:300px;                                 /* 宽度 */
    border-style:solid dashed dotted double;     /* 顶边实线、右边虚线、底边点线、左边双线 */
    border-width:10px 20px 30px 40px;            /* 顶边 10px、右边 20px、底边 30px、左边 40px */
    border-color:red blue green yellow;          /* 顶边红色、右边蓝色、底边绿色、左边黄色 */
}
</style>
<img src="images/1.png" />
```

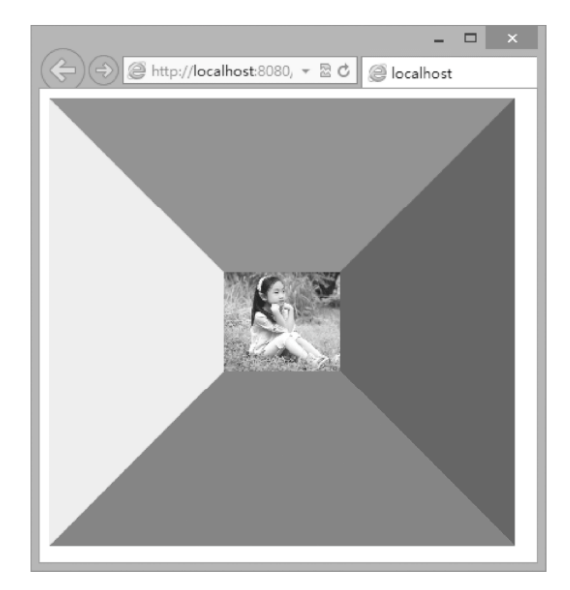


图 4.4 定义各边边框的样式效果



图 4.5 自定义各边边框的样式效果



如果各边样式相同，使用 border 会更方便设计。例如，下面代码定义各边样式为红色实线框，宽度为 20 像素。



```
div {
    width:400px;                    /* 宽度 */
    height:200px;                   /* 高度 */
    border:solid 20px red;          /* 边框样式 */
}
```

在上面代码中，border 属性中的 3 个值分别表示边框样式、边框宽度和边框颜色，它们没有先后顺序，可以任意调整顺序。



视频讲解

4.1.3 定义图像不透明度

在 CSS3 中，使用 opacity 属性可以设计图像的不透明度。该属性的基本用法如下。

```
opacity:0~1;
```

取值范围为 0～1，数值越小，透明度越高，0 为完全透明，而 1 表示完全不透明。

早期 IE 浏览器使用 CSS 滤镜定义透明度，基本用法如下。

```
filter:alpha(opacity=0~100);
```

取值范围为 0～100，数值越小，透明度越高，0 为完全透明，100 表示完全不透明。

【示例】在本示例中，先定义一个透明度样式类，然后把它应用到一个图像中，并与原图进行比较，演示效果如图 4.6 所示。

```
<style type="text/css">
img { width:300px;}
.opacity {/* 透明度样式类 */
    opacity: 0.3;                   /* 标准用法 */
    filter:alpha(opacity=30);       /* 兼容 IE 早期版本浏览器 */
    -moz-opacity:0.3;               /* 兼容 Firefox 浏览器 */
}
</style>
<img src="images/1.png" title="图像不透明度" />
<img class="opacity" src="images/1.png" title="图像透明度为 0.3" />
```



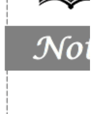




图 4.6　图像透明度演示效果



4.1.4　定义圆角图像

CSS3 新增了 border-radius 属性，使用它可以设计圆角样式。该属性用法如下。

```
border-radius:none | <length>{1,4} [ / <length>{1,4} ]?;
```

border-radius 属性初始值为 none，适用于所有元素，除了 border-collapse 属性值为 collapse 的 table 元素。取值简单说明如下。

☑　none：默认值，表示元素没有圆角。

☑　<length>：由浮点数字和单位标识符组成的长度值，不可为负值。

为了方便定义元素的 4 个顶角圆角，border-radius 属性派生了以下 4 个子属性。

☑　border-top-right-radius：定义右上角的圆角。

☑　border-bottom-right-radius：定义右下角的圆角。

☑　border-bottom-left-radius：定义左下角的圆角。

☑　border-top-left-radius：定义左上角的圆角。

> **提示：** border-radius 属性可包含两个参数值：第 1 个值表示圆角的水平半径，第 2 个值表示圆角的垂直半径，两个参数值通过斜线分隔。如果仅包含 1 个参数值，则第 2 个值与第 1 个值相同，它表示这个角是一个 1/4 圆角。如果参数值中包含 0，则这个角就是矩形，不会显示为圆角。

【示例】在本示例中，分别设计两个圆角类样式，第 1 个类 r1 为固定 12 像素的圆角，第 2 个类 r2 为弹性取值 50%的椭圆圆角，然后分别应用到不同的图像上。演示效果如图 4.7 所示。

```
<style type="text/css">
img { width:300px;border:solid 1px #eee;}
.r1 { border-radius:12px; }
```




```
.r2 { border-radius:50%;}
</style>
<img class="r1" src="images/1.png" title="圆角图像" />
<img class="r2" src="images/1.png" title="椭圆图像" />
```

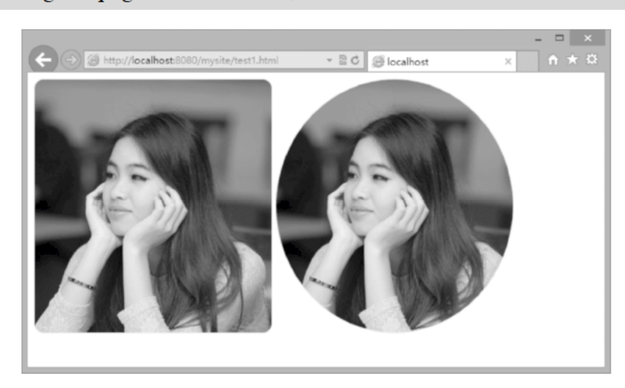


图 4.7　圆角图像演示效果



视频讲解

4.1.5　定义阴影图像

CSS3 新增了 box-shadow 属性，该属性可以定义阴影效果。该属性的用法如下。

```
box-shadow:none | <shadow> [ , <shadow> ]*;
```

box-shadow 属性的初始值是 none，该属性适用于所有元素。取值简单说明如下。

☑　none：默认值，表示元素没有阴影。

☑　<shadow>：该属性值可以使用公式表示为 inset && [<length>{2,4} && <color>?]，其中 inset 表示设置阴影的类型为内阴影，默认为外阴影。<length>是由浮点数字和单位标识符组成的长度值，可取正负值，用来定义阴影水平偏移、垂直偏移，以及阴影大小（即阴影模糊度）、阴影扩展。<color>表示阴影颜色。

提示： 当不设置阴影类型时，默认为投影效果，当设置为 inset 时，则阴影效果为内阴影。X 轴偏移和 Y 轴偏移定义阴影的偏移距离。阴影大小、阴影扩展和阴影颜色是可选值，默认为黑色实影。box-shadow 属性值必须设置阴影的偏移值，否则没有效果。如果需要定义阴影，不需要偏移，此时可以定义阴影偏移为 0，这样才可以看到阴影效果。

【示例 1】 在本示例中，设计一个阴影类样式，定义圆角、阴影显示，设置圆角大小为 8 像素，阴影显示在右下角，模糊半径为 14 像素，然后分别应用于第二幅图像上。演示效果如图 4.8 所示。

```
<style type="text/css">
img { width:300px; margin:6px;}
.r1 {
    border-radius:8px;
    -moz-box-shadow:8px 8px 14px #06C;          /*兼容 Gecko 引擎*/
```






```
        -webkit-box-shadow:8px 8px 14px #06C; /*兼容 Webkit 引擎*/
        box-shadow:8px 8px 14px #06C;          /*标准用法*/
    }
</style>
<img src="images/1.png" title="无阴影图像" />
<img class="r1" src="images/1.png" title="阴影图像" />
```



图 4.8　阴影图像演示效果

【示例 2】box-shadow 属性用法比较灵活，可以设计叠加阴影特效。例如，在上面的示例中，修改类样式 r1 的代码如下，通过多组参数值定义的渐变阴影效果如图 4.9 所示。

```
img { width:300px; margin:6px;}
.r1 {
    border-radius:12px;
    box-shadow:-10px 0 12px red,
      10px 0 12px blue,
      0 -10px 12px yellow,
      0 10px 12px green;
}
```

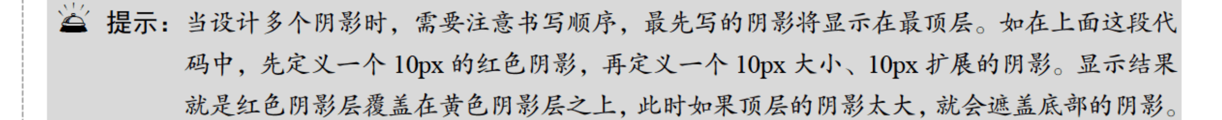


图 4.9　设计图像多层阴影效果





Note



提示：当设计多个阴影时，需要注意书写顺序，最先写的阴影将显示在最顶层。如在上面这段代码中，先定义一个 10px 的红色阴影，再定义一个 10px 大小、10px 扩展的阴影。显示结果就是红色阴影层覆盖在黄色阴影层之上，此时如果顶层的阴影太大，就会遮盖底部的阴影。

4.2 设计背景图像样式

下面学习如何使用 CSS 设计背景图像的显示样式。



视频讲解

4.2.1 定义背景图像

在 CSS 中可以使用 background-image 属性来定义背景图像。具体用法如下。

```
background-image：none | <url>
```

默认值为 none，表示无背景图；<url>表示使用绝对或相对地址指定背景图像。

提示：GIF 格式的图像可以设计动画、透明背景等，图像小巧，而 JPG 格式的图像具有更丰富的颜色数，图像品质相对要好，PNG 类型综合了 GIF 和 JPG 两种图像的优点。

【示例】如果背景包含透明区域的 GIF 或 PNG 格式图像，则被设置为背景图像时，这些透明区域依然被保留。在本示例中，先为网页定义背景图像，然后为段落文本定义透明的 GIF 背景图像。显示效果如图 4.10 所示。

```
<style type="text/css">
html, body, p{ height:100%;}
body {background-image:url(images/bg.jpg);}
p { background-image:url(images/ren.png);}
</style>
<p></p>
```

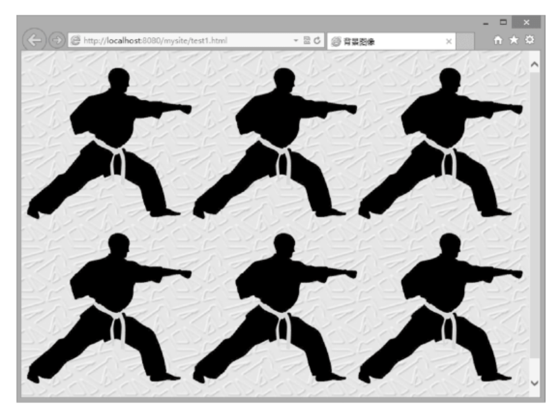


图 4.10　透明背景图像的显示效果







4.2.2　定义显示方式

CSS 使用 background-repeat 属性控制背景图像的显示方式。具体用法如下。

```
background-repeat：repeat-x | repeat-y | [repeat | space | round | no-repeat]{1,2}
```

取值说明如下。

- ☑ repeat-x：背景图像在横向上平铺。
- ☑ repeat-y：背景图像在纵向上平铺。
- ☑ repeat：背景图像在横向和纵向平铺。
- ☑ round：背景图像自动缩放直到适应且填满整个容器，仅 CSS3 支持。
- ☑ space：背景图像以相同的间距平铺且填满整个容器或某个方向，仅 CSS3 支持。
- ☑ no-repeat：背景图像不平铺。

【示例】本示例设计了一个公司公告栏，其中宽度是固定的，但是高度可能会根据正文内容进行动态调整，为了适应这种设计需要，不妨利用垂直平铺来进行设计。

第 1 步，把“公司公告”栏目分隔为上、中、下 3 块，设计<div id="call_tit">和<div id="call_btm">为固定高度，而中间块<div id="call_mid">可以随时调整高度。设计的结构如下。

```
<div id="call">
    <div id="call_tit">公司公告</div >
    <div id="call_mid"></div >
    <div id="call_btm"></div >
</div>
```

第 2 步，所实现的样式表如下，最后调整中间块元素的高度，以形成不同高度的公告牌，演示效果如图 4.11 所示。

```
<style type="text/css">
#call {
    width:218px;                              /* 固定宽度 */
    font-size:14px;                           /* 字体大小 */
}
#call_tit {
    background:url(images/call_top.gif);      /* 头部背景图像 */
    background-repeat:no-repeat;              /* 不平铺显示 */
    height:43px;                              /* 固定高度，与背景图像高度一致 */
    color:#fff;                               /* 白色标题 */
    font-weight:bold;                         /* 粗体 */
    text-align:center;                        /* 居中显示 */
    line-height:43px;                         /* 标题垂直居中 */
}
```




Note

```
#call_mid {
    background-image:url(images/call_mid.gif);    /* 背景图像 */
    background-repeat:repeat-y;                    /* 垂直平铺 */
    height:160px;                                  /* 可自由设置的高度 */
}
#call_btm {
    background-image:url(images/call_btm.gif);    /* 底部背景图像 */
    background-repeat:no-repeat;                   /* 不平铺显示 */
    height:11px;                                   /* 固定高度，与背景图像高度一致 */
}
```

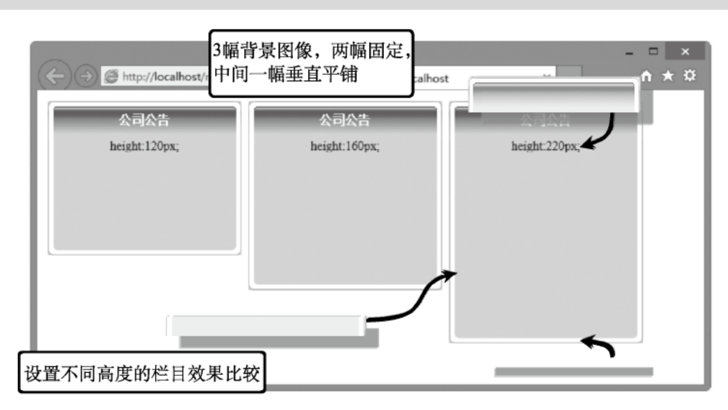


图 4.11　背景图像垂直平铺示例模拟效果



视 频 讲 解

4.2.3　定义显示位置

在默认情况下，背景图像显示在元素的左上角，并根据不同方式执行不同的显示效果。为了更好地控制背景图像的显示位置，CSS 定义了 background-position 属性来精确定位背景图像。

background-position 属性取值包括两个值，分别用来定位背景图像的 x 轴、y 轴坐标，取值单位没有限制。具体用法如下。

```
background-position: [ left | center | right | top | bottom | <percentage> | <length> ] | [ left | center | right | <percentage> | <length> ] [ top | center | bottom | <percentage> | <length> ] | [ center | [ left | right ] [ <percentage> | <length> ]? ] && [ center | [ top | bottom ] [ <percentage> | <length> ]? ]
```

默认值为 0% 0%，等效于 left top。

【示例】本示例利用 4 个背景图像拼接起来组成一个栏目版块，这些背景图像分别被定位到栏目的 4 条边上，形成一个圆角矩形，并富有立体感，效果如图 4.12 所示。





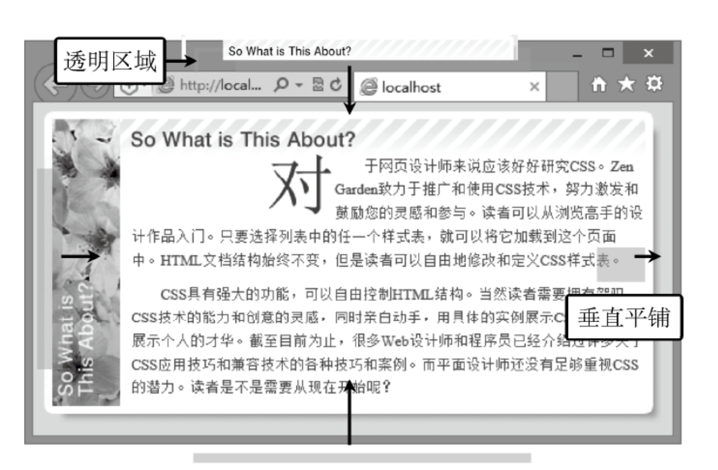


图 4.12　背景图像定位综合应用

示例所用到的 HTML 结构代码如下。

```
<div id="explanation">
    <h3><span>这是什么？</span></h3>
    <p class="p1"><span><span class="first">对</span>于网页设计师来说应该好好研究<acronym title="cascading style sheets">CSS</acronym>。Zen Garden 致力于推广和使用 CSS 技术，努力激发和鼓励您的灵感和参与。读者可以从浏览高手的设计作品入门。只要选择列表中的任一个样式表，就可以将它加载到这个页面中。<acronym title="hypertext markup language">HTML</acronym>文档结构始终不变，但是读者可以自由地修改和定义<acronym title="cascading style sheets">CSS</acronym>样式表。</span></p>
    <p class="p2"><span><acronym title="cascading style sheets">CSS</acronym>具有强大的功能，可以自由控制 HTML 结构。当然读者需要拥有驾驭 CSS 技术的能力和创意的灵感，同时亲自动手，用具体的实例展示 CSS 的魅力，展示个人的才华。截至目前，很多 Web 设计师和程序员已经介绍过许多关于 CSS 应用技巧和兼容技术的各种技巧和案例。而平面设计师还没有足够重视 CSS 的潜力。读者是不是需要从现在开始呢？</span></p>
</div>
```

根据这个 HTML 结构所设计的 CSS 样式表如下，请注意背景图像的定位方法。

```
<STYLE type="text/css">
body { /* 定义网页背景色、居中显示、字体颜色 */
    background:#DFDFDF; text-align:center; color:#454545;
}
p, h3 { margin:0; padding:0; }                      /* 清除段落和标题的默认边距 */
#explanation {
    background-color:#ffffff;                       /* 白色背景，填充所有区域 */
    background-image:url(images/img_explanation.jpg);   /* 指定背景图像 */
    background-position:left bottom;                /* 定位背景图像位于左下角 */
```






```
        background-repeat:repeat-y;                          /* 在垂直方向上平铺背景图像 */
        width:546px;                                         /* 固定栏目宽度 */
        margin:0 auto;                                       /* 栏目居中显示 */
        font-size:13px; line-height:1.6em; text-indent:2em;  /* 定义栏目内字体属性 */
    }
    #explanation h3 {
        background:url(images/title_explanation.gif) no-repeat;  /* 顶部背景图像，不平铺 */
        height:39px;                                         /* 固定标题栏高度 */
    }
    #explanation h3 span { display:none; }                   /* 隐藏标题栏内信息 */
    #explanation p {                                         /* 定义右侧背景图像，垂直平铺 */
       background:url(images/right_bg.gif) right repeat-y;
       }
    #explanation .p2 span {                                  /* 底部背景图像，不平铺 */
        padding-bottom:20px;                                 /* 增加第 2 段底部内边距，显示背景图像 */
        background:url(images/right_bottom.gif) bottom no-repeat;
    }
    #explanation p span {/* 定义段落文本左侧的内边距，以便显示左侧背景图像 */
        padding:0 15px 10px 77px;
        display:block;                                       /* 定义块状显示，内边距才有效 */
        text-align:left;                                     /* 文本左对齐 */
    }
    #explanation p .first {/* 定义首字下沉特效 */
        font-size:60px; color:#820015; line-height:1em;      /* 字体显示属性 */
        float:left;                                          /* 向左浮动 */
        padding:0;                                           /* 清除上面样式为段落定义的内边距 */
    }
    </STYLE>
```

在上面的样式表中，分别为不同元素定义背景图像，然后通过定位技术把背景图像定位到对应的 4 条边上，并根据需要运用平铺技术实现圆角区域效果。

注意：百分比是最灵活的定位方式，同时也是最难把握的定位单位。

在默认状态下，定位的位置为（0% 0%），定位点是背景图像的左上顶点，定位距离是该点到包含框左上角顶点的距离，即两点重合。

如果定位背景图像为（100% 100%），定位点是背景图像的右下顶点，定位距离是该点到包含框左上角顶点的距离，这个距离等于包含框的宽度和高度。

百分比也可以取负值，负值的定位点是包含框的左上顶点，而定位距离则由图像自身的宽和高





来决定。

CSS 还提供了 5 个关键字：left、right、center、top 和 bottom。这些关键字实际上就是百分比特殊值的一种固定用法。详细列表说明如下。

```
/* 普通用法 */
top left、left top                    = 0% 0%
right top、top right                  = 100% 0%
bottom left、left bottom              = 0% 100%
bottom right、right bottom            = 100% 100%
/* 居中用法 */
center、center center                 = 50% 50%
/* 特殊用法 */
top、top center、center top           = 50% 0%
left、left center、center left        = 0% 50%
right、right center、center right     =100% 50%
bottom、bottom center、center bottom  = 50% 100%
```

4.2.4　定义固定背景



在默认情况下，背景图像能够跟随网页内容上下滚动。可以使用 background-attachment 属性定义背景图像在窗口内固定显示，具体用法如下。

```
background-attachment：fixed | local | scroll
```

默认值为 scroll，具体取值说明如下。

☑ fixed：背景图像相对于浏览器窗体固定。

☑ local：背景图像相对于元素内容固定，也就是说当元素内容滚动时，背景图像也会跟着滚动，此时不管元素本身是否滚动，当元素显示滚动条时才会看到效果。该属性值仅 CSS3 支持。

☑ scroll：背景图像相对于元素固定，也就是说当元素内容滚动时，背景图像不会跟着滚动，因为背景图像总是要跟着元素本身。

【示例】 在本示例中，为<body>标签设置背景图片，且不平铺、固定，这时通过拖动浏览器滚动条，可以看到网页内容在滚动，而背景图片固定显示。

```
<style type="text/css">
body {
    background-image: url(images/bg.jpg);              /* 设置背景图片 */
    background-repeat: no-repeat;                      /* 背景图片不平铺 */
    background-position: left center;                  /* 背景图片的位置 */
    background-attachment: fixed;                      /* 背景图片固定，不随滚动条滚动而滚动 */
```




```
        height: 1200px;                               /* 高度，出现浏览器的滚动条 */
}
#box {float:right; width:400px;}
</style>
<div id="box">
        <h1> 雨巷</h1>
        <h2>戴望舒</h2>
        <pre>
撑着油纸伞，独自
彷徨在悠长、悠长
又寂寥的雨巷，
我希望逢着
一个丁香一样的
结着愁怨的姑娘。
……
        </pre>
</div>
```

页面演示效果如图 4.13 所示。

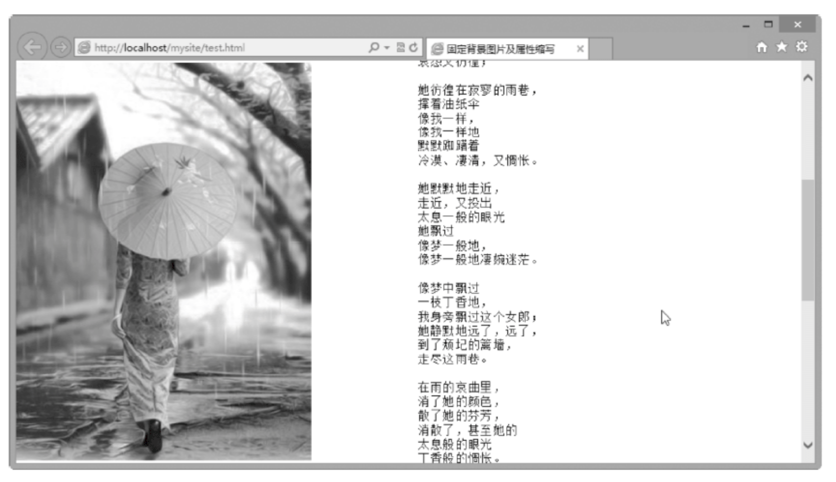


图 4.13　背景图片固定

4.2.5　定义原点



视频讲解

background-origin 属性定义 background-position 属性的定位原点。在默认情况下，background-position 属性总是以元素左上角为坐标原点定位背景图像。使用 background-origin 属性可以改变这种定位方式。该属性的基本语法如下。

```
background-origin:border-box | padding-box | content-box;
```



取值简单说明如下。

☑　border-box：从边框区域开始显示背景。

☑　padding-box：从补白区域开始显示背景，为默认值。

☑　content-box：仅在内容区域显示背景。

【示例】background-origin 属性改善了背景图像定位的方式，更灵活地决定背景图像应该显示的位置。本示例利用 background-origin 属性重设背景图像的定位坐标，以便更好地控制背景图像的显示，演示效果如图 4.14 所示。



图 4.14　设计诗词效果

示例代码如下。

```
<style type="text/css">
div {/*定义包含框的样式*/
    height: 322px;
    width: 780px;
    border: solid 1px red;
    padding: 250px 4em 0;
    /*为了避免背景图像重复平铺到边框区域，应禁止它平铺*/
    background:url(images/p3.jpg) no-repeat;
    /*设计背景图像的定位坐标点为元素边框的左上角*/
    background-origin:border-box;
    /*将背景图像等比缩放到完全覆盖包含框，背景图像有可能超出包含框*/
    background-size:cover;
    overflow:hidden;                    /*隐藏超出包含框的内容*/
}
div h1, div h2{/*定义标题样式*/
    font-size:18px; font-family:"幼圆";
    text-align:center;                  /*水平居中显示*/
```




```
}
div p {/*定义正文样式*/
    text-indent:2em;                 /*首行缩进两个字符*/
    line-height:2em;                 /*增大行高，让正文看起来更疏朗*/
    margin-bottom:2em;               /*调整底部边界，增大段落文本距离*/
}
</style>

<div>
    <h1>念奴娇&#8226;赤壁怀古</h1>
    <h2>苏轼</h2>
    <p>大江东去，浪淘尽，千古风流人物。故垒西边，人道是，三国周郎赤壁。乱石穿空，惊涛拍岸，卷起千堆雪。江山如画，一时多少豪杰。</p>
    <p>遥想公瑾当年，小乔初嫁了，雄姿英发。羽扇纶巾，谈笑间，樯橹灰飞烟灭。故国神游，多情应笑我，早生华发。人生如梦，一尊还酹江月。</p>
</div>
```



视频讲解

4.2.6 定义裁剪区域

background-clip 属性定义背景图像的裁剪区域。该属性的基本语法如下。

```
background-clip:border-box | padding-box | content-box | text;
```

取值简单说明如下。

☑ border-box：从边框区域向外裁剪背景，为默认值。

☑ padding-box：从补白区域向外裁剪背景。

☑ content-box：从内容区域向外裁剪背景。

☑ text：从前景内容（如文字）区域向外裁剪背景。

提示：如果取值为 padding-box，则 background-image 将忽略补白边缘，此时边框区域显示为透明；

如果取值为 border-box，则 background-image 将包括边框区域；

如果取值为 content-box，则 background-image 将只包含内容区域；

如果 background-image 属性定义了多重背景，则 background-clip 属性值可以设置多个值，并用逗号分隔。

如果background-clip属性值为padding-box，background-origin属性值为border-box，且background-position 属性值为"top left"（默认初始值），则背景图左上角将会被截取掉部分。





【示例 1】本示例演示了如何设计背景图像仅在内容区域内显示，演示效果如图 4.15 所示。

```
<style type="text/css">
div {
    height:150px;
    width:300px;
    border:solid 50px gray;
    padding:50px;
    background:url(images/bg.jpg) no-repeat;
    /*将背景图像等比缩放到完全覆盖包含框，背景图像有可能超出包含框*/
    background-size:cover;
    /*将背景图像从 content 区域开始向外裁剪背景*/
    background-clip:content-box;
}
</style>

<div></div>
```

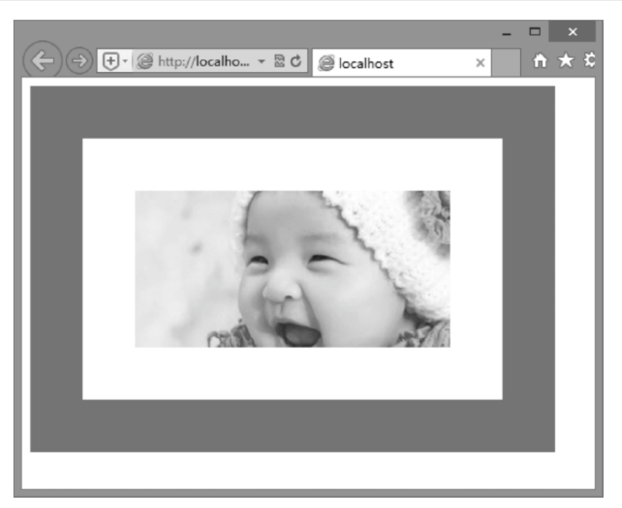


图 4.15　以内容边缘裁切背景图像效果

【示例 2】本示例同时定义 background-clip 和 background-origin 属性值为 content，可以设计比较特殊的按钮样式，演示效果如图 4.16 所示。

```
<style type="text/css">
button {
    height:40px;                    /*固定包含框大小*/
    width:150px;
    padding:1px;                    /*在内容区留点空隙*/
```




Note

```
        cursor:pointer;                         /*定义手形指针样式*/
        color:#fff;                             /*白色字体*/
        /*设计立体边框样式*/
        border:3px double #95071b;
        border-right-color:#650513;
        border-bottom-color:#650513;
        /*为了避免背景图像重复平铺到边框区域，应禁止它平铺 */
        background:url(images/img6.jpg) no-repeat;
        /*设计背景图像的定位坐标点为元素内容区域的左上角*/
        background-origin:content-box;
        /*设计以内容区域的边缘来裁切背景图像*/
        background-clip:content-box;
    }
    </style>

    <button>导航按钮 >></button>
```



图 4.16　设计按钮效果


视频讲解

4.2.7　定义大小

background-size 可以控制背景图像的显示大小。该属性的基本语法如下。

```
background-size: [ <length> | <percentage> | auto ]{1,2} | cover | contain;
```

取值简单说明如下。

☑　<length>：由浮点数字和单位标识符组成的长度值。不可为负值。

☑　<percentage>：取值范围为 0～100%。不可为负值。

☑　cover：保持背景图像本身的宽高比例，将图片缩放到正好完全覆盖所定义背景的区域。

☑　contain：保持图像本身的宽高比例，将图片缩放到宽度或高度正好适应所定义背景的区域。

background-size 属性初始值为 auto。可以设置 1 个或 2 个值，1 个为必填，1 个为可选。其中第 1 个值用于指定背景图像的 width，第 2 个值用于指定背景图像的 height，如果只设置 1 个值，则第 2 个值默认为 auto。

【示例】设计自适应模块大小的背景图像。借助 image-size 属性自由定制背景图像大小的功能，让背景图像自适应盒子的大小，从而可以设计与模块大小完全适应的背景图像，本示例效果如图 4.17





所示，只要背景图像长宽比与元素长宽比相同，就不用担心背景图像脱节。

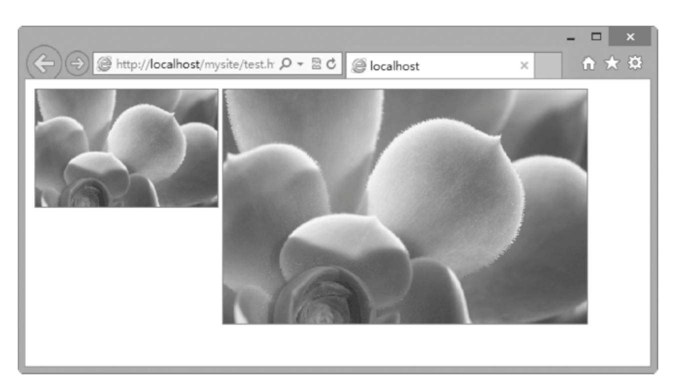




图 4.17　设计背景图像自适应显示

示例完整代码如下。

```
<style type="text/css">
div {
    margin:2px;
    float:left;
    border:solid 1px red;
    background:url(images/img2.jpg) no-repeat center;
    /*设计背景图像完全覆盖元素区域*/
    background-size:cover;
}
/*设计元素大小*/
.h1 { height:120px; width:192px; }
.h2 { height:240px; width:384px; }
</style>
<div class="h1"></div>
<div class="h2"></div>
```

4.2.8　定义多背景图



CSS3 中可以在一个元素里显示多个背景图像，还可以将多个背景图像进行重叠显示，从而使得背景图像中所用素材的调整变得更加容易。

【示例】本示例将用到 8 个背景图像，使用它们分别模拟圆角边框的 4 个顶角和 4 条边。最后通过 CSS 把它们分别固定到元素的边框和顶角上，设计效果如图 4.18 所示。

```
<style type="text/css">
body { text-align: center; }
.roundbox {
```





Note

```
    padding: 2em;
    width: 90%px;
    margin: 0px auto;
    background-image:
        url(images/roundbox1/tl.gif),
        url(images/roundbox1/tr.gif),
        url(images/roundbox1/bl.gif),
        url(images/roundbox1/br.gif),
        url(images/roundbox1/right.gif),
        url(images/roundbox1/left.gif),
        url(images/roundbox1/top.gif),
        url(images/roundbox1/bottom.gif);
    background-repeat:
        no-repeat,
        no-repeat,
        no-repeat,
        no-repeat,
        repeat-y,
        repeat-y,
        repeat-x,
        repeat-x;
    background-position:
        left 0px,
        right 0px,
        left bottom,
        right bottom,
        right 0px,
        0px 0px,
        left 0px,
        left bottom;
    background-color: #66CC33;
    text-align: left;
}
</style>
<div class="roundbox">
    <h3><span>这是什么？ </span></h3>
    <p  class="p1"><span><span  class="first">对</span>于网页设计师来说应该好好研究<acronym
title="cascading style sheets">CSS</acronym>。Zen Garden 致力于推广和使用 CSS 技术，努力激发和鼓励您的灵
```



```
感和参与。您可以从浏览高手的设计作品入门。只要选择列表中的任一个样式表，就可以将它加载到这个页面中。<acronym title="hypertext markup language">HTML</acronym>文档结构始终不变，但是您可以自由地修改和定义<acronym title="cascading style sheets">CSS</acronym>样式表。</span></p>
    </div>
```





图 4.18　定义多背景图像

4.3　定义渐变背景样式

基于 CSS3 的渐变背景与渐变图片相比，最大的优点是便于修改，同时支持无级缩放，过渡更加自然。CSS3 支持线性渐变和径向渐变，下面分别进行介绍。

4.3.1　线性渐变



创建线性渐变，至少需要两种颜色，可选择设置一个起点或一个方向。简明语法格式如下。

```
linear-gradient( angle, color-stop1, color-stop2, …)
```

参数简单说明如下。

☑　angle：用来指定渐变的方向，可以使用角度或者关键字来设置。关键字包括 4 个，说明如下。

- to left：设置渐变从右到左，相当于 270deg。
- to right：设置渐变从左到右，相当于 90deg
- to top：设置渐变从下到上，相当于 0deg
- to bottom：设置渐变从上到下，相当于 180deg。该值为默认值。

提示： 如果创建对角线渐变，可以使用类似于 to top left（从右下到左上）的关键字来实现。

☑　color-stop：用于指定渐变的色点，包括一个颜色值和一个起点位置，颜色值和起点位置以空格分隔。起点位置可以为一个具体的长度值（不可为负值），也可以是一个百分比值，如果是百分比值则参考应用渐变对象的尺寸，最终会被转换为具体的长度值。





Note

【示例 1】本示例为<div id="demo">对象应用了一个简单的线性渐变背景，方向从上到下，颜色由白色到浅灰，效果如图 4.19 所示。

```
<style type="text/css">
#demo {
    width:300px;
    height:200px;
    background: linear-gradient(#fff, #333);
}
</style>
<div id="demo"></div>
```

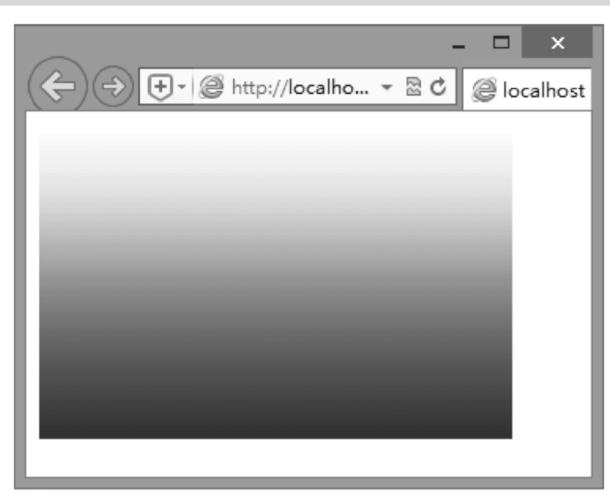


图 4.19　应用简单的线性渐变效果

提示：针对示例 1，用户可以继续尝试做下面的练习，实现通过不同的设置得到相同的设计效果。

☑ 设置一个方向：从上到下，覆盖默认值。

```
linear-gradient(to bottom, #fff, #333);
```

☑ 设置反向渐变：从下到上，同时调整起止颜色位置。

```
linear-gradient(to top, #333, #fff);
```

☑ 使用角度值设置方向。

```
linear-gradient(180deg, #fff, #333);
```

☑ 明确起止颜色的具体位置，覆盖默认值。

```
linear-gradient(to bottom, #fff 0%, #333 100%);
```

【示例 2】本示例演示了从左边开始的线性渐变。起点是红色，慢慢过渡到蓝色，效果如图 4.20





所示。

```
<style type="text/css">
#demo {
    width:300px; height:200px;
    background: -webkit-linear-gradient(left, red , blue);      /* Safari 5.1 - 6.0 */
    background: -o-linear-gradient(left, red, blue);            /* Opera 11.1 - 12.0 */
    background: -moz-linear-gradient(left, red, blue);          /* Firefox 3.6 - 15 */
    background: linear-gradient(to right, red , blue);          /* 标准语法 */
}
</style>
<div id="demo"></div>
```

注意，第一个参数值渐变方向的设置不同。

【示例 3】通过指定水平和垂直的起始位置来设计对角渐变。本示例演示了从左上角到右下角的线性渐变，起点是红色，慢慢过渡到蓝色，效果如图 4.21 所示。

```
#demo {
    width:300px; height:200px;
    background: -webkit-linear-gradient(left top, red , blue);      /* Safari 5.1 - 6.0 */
    background: -o-linear-gradient(left top, red, blue);            /* Opera 11.1 - 12.0 */
    background: -moz-linear-gradient(left top, red, blue);          /* Firefox 3.6 - 15 */
    background: linear-gradient(to bottom right, red , blue);       /* 标准语法 */
}
```

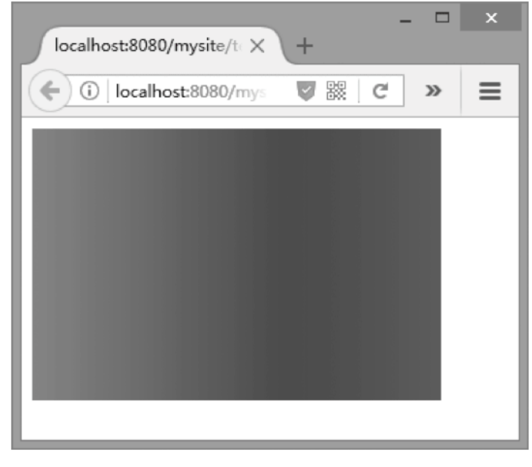


图 4.20　设计从左到右的线性渐变效果

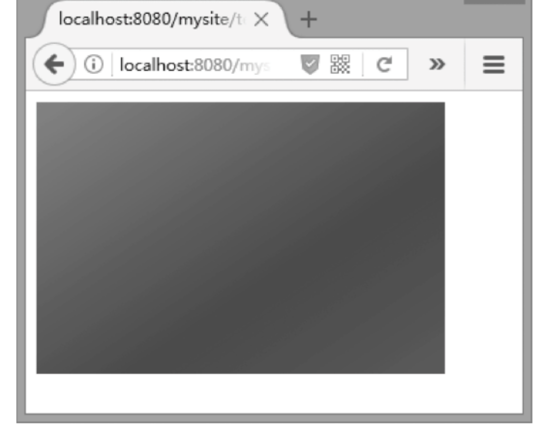


图 4.21　设计对角线性渐变效果

【示例 4】通过指定具体的角度值，可以设计更多渐变方向。本示例演示了从上到下的线性渐变，起点是红色，慢慢过渡到蓝色，效果如图 4.22 所示。





```
#demo {
    width:300px; height:200px;
    background: -webkit-linear-gradient(-90deg, red, blue);    /* Safari 5.1 - 6.0 */
    background: -o-linear-gradient(-90deg, red, blue);         /* Opera 11.1 - 12.0 */
    background: -moz-linear-gradient(-90deg, red, blue);       /* Firefox 3.6 - 15 */
    background: linear-gradient(180deg, red, blue);            /* 标准语法 */
}
```

【补充】

渐变角度是指垂直线和渐变线之间的角度，逆时针方向计算。例如，0deg 将创建一个从下到上的渐变，90deg 将创建一个从左到右的渐变。注意，渐变起点以负 y 轴为参考。

但是，很多浏览器（如 Chrome、Safari、Firefox 等）使用旧的标准：渐变角度是指水平线和渐变线之间的角度，逆时针方向计算。例如，0deg 将创建一个从左到右的渐变，90deg 将创建一个从下到上的渐变。注意，渐变起点以负 x 轴为参考。

兼容公式如下。

```
90 - x = y
```

其中，x 为标准角度，y 为非标准角度。

【示例 5】设置多个色点。本示例定义从上到下的线性渐变，起点是红色，慢慢过渡到绿色，再慢慢过渡到蓝色，效果如图 4.23 所示。

```
#demo {
    width:300px; height:200px;
    background: -webkit-linear-gradient(red, green, blue);    /* Safari 5.1 - 6.0 */
    background: -o-linear-gradient(red, green, blue);         /* Opera 11.1 - 12.0 */
    background: -moz-linear-gradient(red, green, blue);       /* Firefox 3.6 - 15 */
    background: linear-gradient(red, green, blue);            /* 标准语法 */
}
```

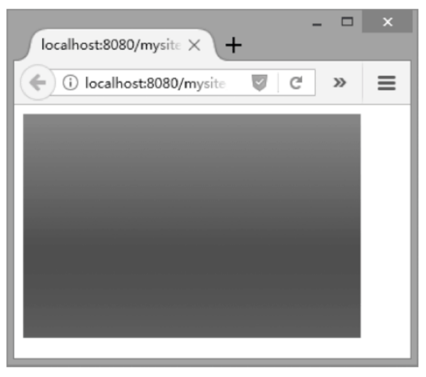


图 4.22　设计从上到下的渐变效果

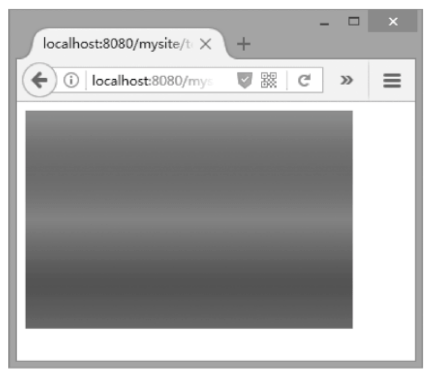


图 4.23　设计多色线性渐变效果





【示例 6】设置色点位置。本示例定义从上到下的线性渐变，起点是黄色，快速过渡到蓝色，再慢慢过渡到绿色，效果如图 4.24 所示。

```
#demo {
    width:300px; height:200px;
    background: -webkit-linear-gradient(yellow, blue 20%, #0f0);          /* Safari 5.1 - 6.0 */
    background: -o-linear-gradient(yellow, blue 20%, #0f0);               /* Opera 11.1 - 12.0 */
    background: -moz-linear-gradient(yellow, blue 20%, #0f0);             /* Firefox 3.6 - 15 */
    background: linear-gradient(yellow, blue 20%, #0f0);                  /*  标准语法 */
}
```

【示例 7】CSS3 渐变支持透明度设置，可用于创建减弱变淡的效果。本示例演示了从左边开始的线性渐变。起点是完全透明，起点位置为 30%，慢慢过渡到完全不透明的红色，为了更清晰地看到半透明效果，增加了一层背景图像进行衬托，演示效果如图 4.25 所示。

```
#demo {
    width:300px; height:200px;
    /* Safari 5.1 - 6 */
    background: -webkit-linear-gradient(left,rgba(255,0,0,0) 30%,rgba(255,0,0,1)),url(images/bg.jpg);
    /* Opera 11.1 - 12*/
    background: -o-linear-gradient(left,rgba(255,0,0,0) 30%,rgba(255,0,0,1)),url(images/bg.jpg);
    /* Firefox 3.6 - 15*/
    background: -moz-linear-gradient(left,rgba(255,0,0,0) 30%,rgba(255,0,0,1)),url(images/bg.jpg);
    /*  标准语法 */
    background: linear-gradient(to right, rgba(255,0,0,0) 30%, rgba(255,0,0,1)),url(images/bg.jpg);
    background-size:cover;                /*  背景图像完全覆盖 */
}
```

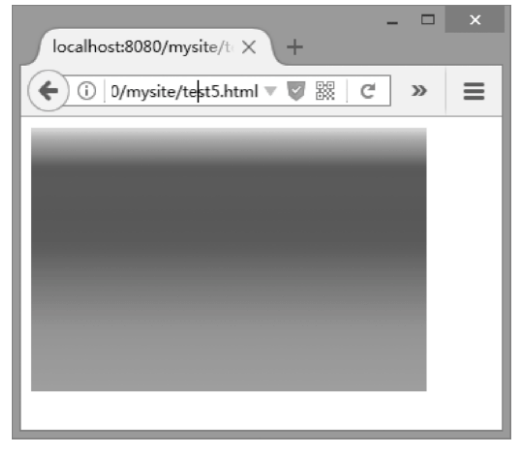


图 4.24　设计多色线性渐变效果

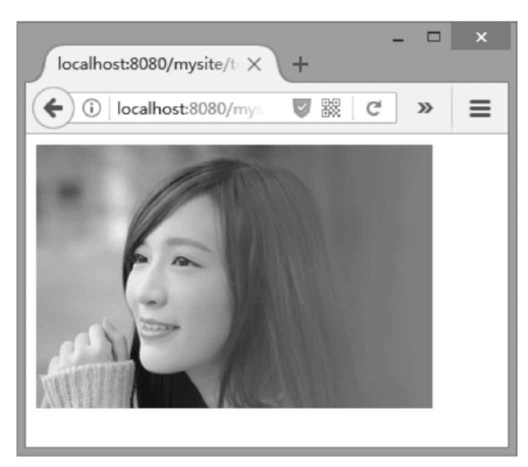


图 4.25　设计半透明线性渐变效果



> 提示：为了添加透明度，可以使用 rgba() 或 hsla()函数来定义色点。rgba()或 hsla()函数中最后一个参数可以是从 0 到 1 的值，它定义了颜色的透明度：0 表示完全透明，1 表示完全不透明。



Note



视频讲解

4.3.2 径向渐变

创建一个径向渐变，也至少需要定义两种颜色，同时可以指定渐变的中心点位置、形状类型（圆形或椭圆形）和半径大小。简明语法格式如下。

```
radial-gradient(shape size at position, color-stop1, color-stop2, …);
```

参数简单说明如下。

☑ shape：用来指定渐变的类型，包括 circle（圆形）和 ellipse（椭圆）两种。

☑ size：如果类型为 circle，指定一个值设置圆的半径；如果类型为 ellipse，指定两个值分别设置椭圆的 x 轴和 y 轴半径。取值包括长度值、百分比、关键字。关键字说明如下。

- closest-side：指定径向渐变的半径长度为从中心点到最近的边。
- closest-corner：指定径向渐变的半径长度为从中心点到最近的角。
- farthest-side：指定径向渐变的半径长度为从中心点到最远的边。
- farthest-corner：指定径向渐变的半径长度为从中心点到最远的角。

☑ position：用来指定中心点的位置。如果提供两个参数，第 1 个表示 x 轴坐标，第 2 个表示 y 轴坐标；如果只提供 1 个值，第 2 个值默认为 50%，即 center。取值可以是长度值、百分比或者关键字，关键字包括 left（左侧）、center（水平居中）、right（右侧）、top（顶部）、center（垂直居中）、bottom（底部）。

注意，position 值位于 shape 和 size 值后面。

☑ color-stop：用于指定渐变的色点。包括一个颜色值和一个起点位置，颜色值和起点位置以空格分隔。起点位置可以为一个具体的长度值（不可为负值），也可以是一个百分比值，如果是百分比值则参考应用渐变对象的尺寸，最终会被转换为具体的长度值。

【示例 1】在默认情况下，渐变的中心是 center（对象中心点），渐变的形状是 ellipse（椭圆形），渐变的大小是 farthest-corner（表示到最远的角落）。本示例仅为 radial-gradient()函数设置 3 个颜色值，则它将按默认值绘制径向渐变效果，如图 4.26 所示。

```
<style type="text/css">
#demo {
    height:200px;
    background: -webkit-radial-gradient(red, green, blue);        /* Safari 5.1 - 6.0 */
    background: -o-radial-gradient(red, green, blue);             /* Opera 11.6 - 12.0 */
    background: -moz-radial-gradient(red, green, blue);           /* Firefox 3.6 - 15 */
    background: radial-gradient(red, green, blue);                /*  标准语法  */
}
</style>
<div id="demo"></div>
```



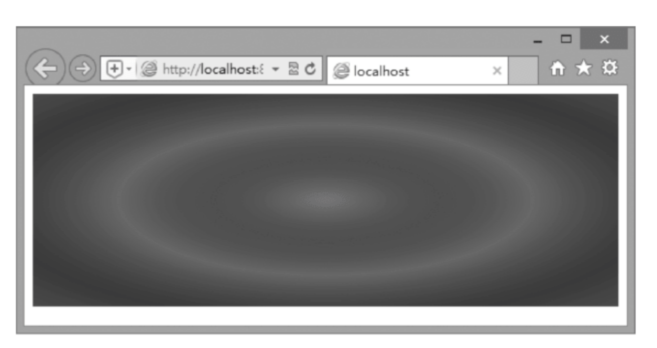


图 4.26　设计简单的径向渐变效果



提示：针对示例 1，用户可以继续尝试做下面的练习，实现通过不同的设置，得到相同的设计效果。

☑ 设置径向渐变形状类型，默认值为 ellipse。

```
background: radial-gradient(ellipse, red, green, blue);
```

☑ 设置径向渐变中心点坐标，默认为对象中心点。

```
background: radial-gradient(ellipse at center 50%, red, green, blue);
```

☑ 设置径向渐变大小，这里定义填充整个对象。

```
background: radial-gradient(farthest-corner, red, green, blue);
```

【示例 2】本示例演示了色点不均匀分布的径向渐变，效果如图 4.27 所示。

```
<style type="text/css">
#demo {
    height:200px;
    background: -webkit-radial-gradient(red 5%, green 15%, blue 60%);      /* Safari 5.1 - 6.0 */
    background: -o-radial-gradient(red 5%, green 15%, blue 60%);           /* Opera 11.6 - 12.0 */
    background: -moz-radial-gradient(red 5%, green 15%, blue 60%);         /* Firefox 3.6 - 15 */
    background: radial-gradient(red 5%, green 15%, blue 60%);              /* 标准语法 */
}
</style>
<div id="demo"></div>
```

【示例 3】shape 参数定义了形状，取值包括 circle 和 ellipse，其中 circle 表示圆形，ellipse 表示椭圆形，默认值是 ellipse。本示例设计圆形径向渐变，效果如图 4.28 所示。

```
#demo {
    height:200px;
    background: -webkit-radial-gradient(circle, red, yellow, green);      /* Safari 5.1 - 6.0 */
```




Note

```
    background: -o-radial-gradient(circle, red, yellow, green);          /* Opera 11.6 - 12.0 */
    background: -moz-radial-gradient(circle, red, yellow, green);        /* Firefox 3.6 - 15 */
    background: radial-gradient(circle, red, yellow, green);             /*  标准语法  */
}
```



图 4.27　设计色点不均匀分布的径向渐变效果

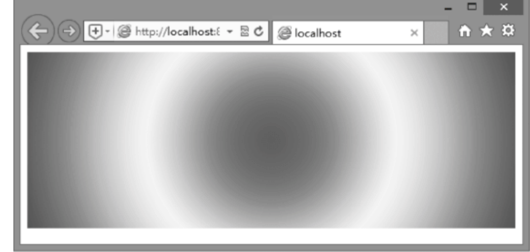


图 4.28　设计圆形径向渐变效果

【示例 4】本示例设计径向渐变的半径为从圆心到离圆心最近的边，如图 4.29 所示。

```
#demo {
    height:200px;
    /* Safari 5.1 - 6.0 */
    background: -webkit-radial-gradient(60% 55%, closest-side,blue,green,yellow,black);
    /* Opera 11.6 - 12.0 */
    background: -o-radial-gradient(60% 55%, closest-side,blue,green,yellow,black);
    /* Firefox 3.6 - 15 */
    background: -moz-radial-gradient(60% 55%, closest-side,blue,green,yellow,black);
    /*  标准语法  */
    background: radial-gradient(closest-side at 60% 55%, blue,green,yellow,black);
}
```

注意，radial-gradient()标准函数与各私有函数在设置参数时顺序不同是有区别的。

【示例 5】本示例模拟的是太阳初升的效果，如图 4.30 所示。设计径向渐变中心点位于左下角，半径为最大化显示，定义 3 个色点，第 1 个色点设计太阳效果，第 2 个色点设计太阳余晖，第 3 个色点设计太空，第 1 个色点和第 2 个色点距离为 60 像素。

```
#demo {
    height:200px;
    /* Safari 5.1 - 6.0 */
    background: -webkit-radial-gradient(left bottom, farthest-side, #f00, #f99 60px, #005);
    /* Opera 11.6 - 12.0 */
    background: -o-radial-gradient(left bottom, farthest-side, #f00, #f99 60px, #005);
    /* Firefox 3.6 - 15 */
    background: -moz-radial-gradient(left bottom, farthest-side, #f00, #f99 60px, #005);
```






```
        /* 标准语法 */
        background: radial-gradient(farthest-side at left bottom, #f00, #f99 60px, #005);
}
```

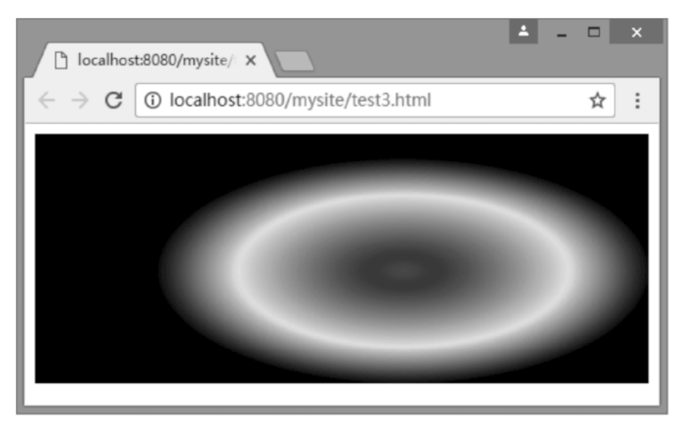


图 4.29 设计最小限度的径向渐变效果

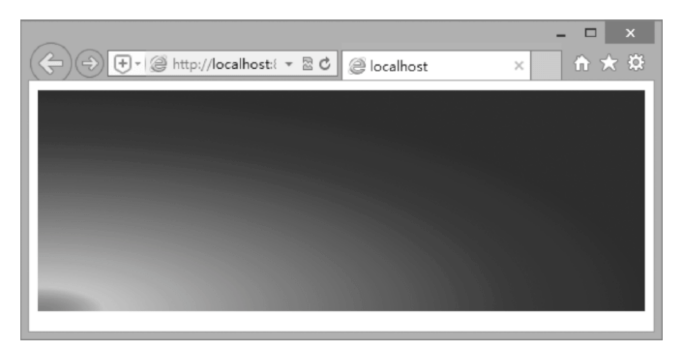


图 4.30 模拟太阳初升效果

【示例 6】本示例模拟太阳旗效果，如图 4.31 所示。设计径向渐变中心点位于对象中央，定义两个色点，第 1 个色点设计太阳效果，第 2 个色点设计背景，两个色点位置相同。

```
<style type="text/css">
body { background:hsla(207,59%,78%,1.00) }
#demo {
        height:200px;
        width:300px;
        margin:auto;
        /* Safari 5.1 - 6.0 */
        background: -webkit-radial-gradient(center, circle, #f00 50px, #fff 50px);
        /* Opera 11.6 - 12.0 */
        background: -o-radial-gradient(center, circle, #f00 50px, #fff 50px);
        /* Firefox 3.6 - 15 */
```




```
        background: -moz-radial-gradient(center, circle, #f00 50px, #fff 50px);
        /* 标准语法 */
        background: radial-gradient(circle  at center, #f00 50px, #fff 50px);
    }
    </style>
    <div id="demo"></div>
```

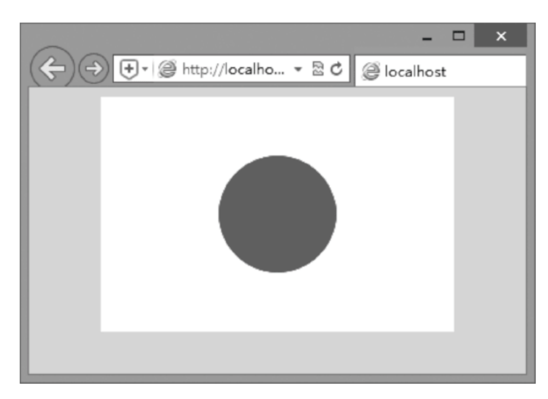


图 4.31　模拟太阳旗效果

4.4　案 例 实 战

图像在网页设计中的作用很重要，恰当地使用背景图像能够设计出非常漂亮、精致的页面效果。下面将结合多个案例介绍图像在网页制作中的魅力。


视 频 讲 解

4.4.1　设计图文混排版式

在网页中经常会看到图文混排的版式，不管是单图还是多图，也不管是简单的文字介绍还是大段正文，图文版式的处理方式是比较简单的。在本节案例中所展示的图文混排效果，主要是文字围绕在图片的旁边进行显示。

【操作步骤】

第 1 步，启动 Dreamweaver，新建网页，保存为 test.html，在<body>标签内输入以下代码，代码中省略部分段落文本，读者可以参考本节案例源代码进行补全。

```
<div class="pic_news">
    <h2>儿童节的来历</h2>
    <p><img src="images/1.jpg" alt="" /><p>
    <p>六一儿童节，也叫“六一国际儿童节”，每年 6 月 1 日举行，是全世界少年儿童的节日。</p>
    <p>……</p>
```





```
</div>
```

第 2 步，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后输入下面样式，设置图片的属性，将其控制到内容区域的左上角。

```
<style type="text/css">
.pic_news { width: 500px; /*  控制内容区域的宽度，根据实际情况考虑，也可以不需要  */ }
.pic_news h2 {
    font-family: "隶书";
    font-size: 24px;
    text-align: center;
    }
.pic_news img {
    float: left; /*  使图片旁边的文字产生浮动效果  */
    margin-right: 16px;
    margin-bottom: 16px;
    height: 250px;
    }
.pic_news p { text-indent:2em; /*  首行缩进 2 个字符  */
    }
</style>
```

简单几行 CSS 样式代码就能实现图文混排的页面效果，其显示效果如图 4.32 所示，其中重点内容就是将图片设置为浮动，float:left 就是将图片向左浮动。



图 4.32　图文混排的页面效果



4.4.2 设计按钮





本例设计的纯 CSS 按钮具有如下特点。

☑ 不需要图片和 JavaScript。

☑ 支持 3 种按钮状态：正常、悬停和激活。

☑ 可以应用到任何 HTML 元素，如 a、input、button、span、div、p、h3 等。

☑ 安全兼容不支持 CSS3 的浏览器，如果不兼容 CSS3，则显示没有渐变和阴影的普通按钮。

本案例设计的按钮效果如图 4.33 所示。按钮在正常状态下有边框的渐变和阴影，在鼠标经过时按钮会显示比较暗的渐变效果，当按下鼠标时会翻转渐变，并显示一个像素的下沉效果，按钮字体颜色加深。



图 4.33 设计精致的按钮

本例主要代码如下。

```
<style type="text/css">
body {
    background:#ededed;
    margin: 30px auto;
    color: #999;
}
.button { /* 定义渐变按钮样式类 */
    display: inline-block;
    /*zoom 和 *display 属性都为了兼容 IE7，使其具有 display:inlineblock 特性*/
    zoom: 1;
    *display: inline;
    vertical-align: baseline;          /* 垂直基线对齐 */
    margin: 0 2px;
    outline: none;                     /* 取消表单按钮控件默认的轮廓线 */
    cursor: pointer;                   /* 鼠标经过时显示手形指针 */
    text-align: center;                /* 文本水平居中 */
    text-decoration: none;             /* 取消 a 元素的下划线 */
    font: 14px/100% Arial, Helvetica, sans-serif;
```





```
        padding: .5em 2em .55em;            /* 调整按钮大小 */
        /*设计按钮圆角、盒子阴影和文本阴影特效*/
        text-shadow: 0 1px 1px rgba(0, 0, 0, .3);
        border-radius: .5em;
        box-shadow: 0 1px 2px rgba(0, 0, 0, .2);
}
.button:hover { text-decoration: none; }
.button:active {
        position: relative;
        top: 1px;
}
.bigrounded { /* 定义大圆角样式类 */
        border-radius: 2em;
}
.medium { /* 定义大按钮样式类 */
        font-size: 12px;
        padding: .4em 1.5em .42em;
}
.small { /* 定义小按钮样式类 */
        font-size: 11px;
        padding: .2em 1em .275em;
}
/* 设计颜色样式类：黑色风格的按钮 */
/* 通过设计不同颜色样式类，可以设计不同风格的按钮效果 */
.black { /* 黑色样式类 */
        color: #d7d7d7;
        border: solid 1px #333;
        background: #333;
        background: linear-gradient(to top, #666, #000);
}
.black:hover { /* 黑色鼠标经过样式类 */
        background: #000;
        background: linear-gradient(to top, #444, #000);
}
.black:active { /* 黑色鼠标激活样式类 */
        color: #666;
        background: linear-gradient(to top, #000, #444);
}
```






```
</style>
<div>
    <a href="#" class="button black">圆角按钮</a>
    <a href="#" class="button black bigrounded">大号椭圆按钮</a>
    <a href="#" class="button black medium">中号按钮</a>
    <a href="#" class="button black small">小号按钮</a> <br />
</div>
```



4.4.3 设计花边框

本例使用 CSS3 多背景设计花边框，使用 background-origin 属性定义仅在内容区域显示背景，使用 background-clip 属性定义背景从边框区域向外裁剪，效果如图 4.34 所示。



图 4.34　设计花边框效果

本例主要代码如下。

```
<style type="text/css">
.demo {
    width: 400px; padding: 30px 30px; border: 20px solid rgba(104, 104, 142,0.5);
    border-radius: 10px;
    color: #f36; font-size: 80px; font-family:"隶书";line-height: 1.5; text-align: center;
      }
.multipleBg {
    background: url("images/bg-tl.png")no-repeat left top, url("images/bg-tr.png")no-repeat right top, url
("images/bg-bl.png")no-repeat left bottom, url("images/bg-br.png") no-repeat right bottom, url("images/bg-repeat.png")
repeat left top;
    /*改变背景图片的 position 起始点，四朵花都是 border 边缘处起，而平铺背景是在 paddin 内边缘起*/
    background-origin: border-box, border-box, border-box, border-box, padding-box;
    /*控制背景图片的显示区域，所有背景图片超过 border 外边缘都将被剪切掉*/
    background-clip: border-box;
}
```



```
</style>

<div class="demo multipleBg">恭喜发财</div>
```

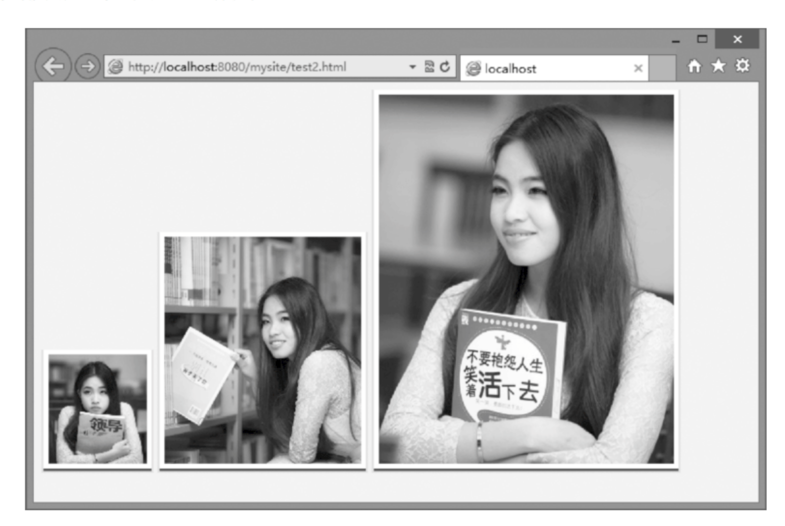




4.4.4　设计图片镶边特效

本例使用背景图像为照片设计镶边效果。这里为 img 元素定义一个默认的阴影样式，这样当在网页中插入一个图像时，它会自动显示为阴影效果，如图 4.35 所示。

其实定义这样的默认样式比较简单，首先需要在图像编辑器中设计一个 4 像素高、1 像素宽的渐变阴影，如图 4.36 所示。



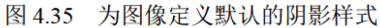


图 4.35　为图像定义默认的阴影样式

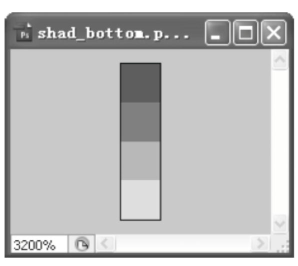


图 4.36　设计一个渐变阴影图像

然后在网页中定义如下样式即可。注意，在定义底边内边距时，考虑到底边阴影背景图像可能要占用 4 个像素的高度，因此要多设置 4 像素。左右两侧的阴影颜色可以根据网页背景色适当调整深浅。

```
<style type="text/css">
body { background: #F0EADA; }
img {
    background: white;                                                    /* 白色背景 */
    padding: 5px 5px 9px 5px;                                             /* 增加内边距 */
    background: white url(images/shad_bottom.gif) repeat-x bottom left;   /* 底边阴影 */
    border-left: 2px solid #dcd7c8;                                       /* 左侧浅阴影 */
    border-right: 2px solid #dcd7c8;                                      /* 右侧浅阴影 */
}
</style>
```





```
<img src="images/1.jpg" width="200">
<img src="images/2.jpg" width="300">
<img src="images/3.jpg" width="400">
```



Note



视频讲解

4.4.5 设计发光的球体

本例使用 CSS3 径向渐变制作圆形球体，主要利用多重背景进行设计，然后使用径向渐变叠加设计球体和发光的光晕效果，如图 4.37 所示。

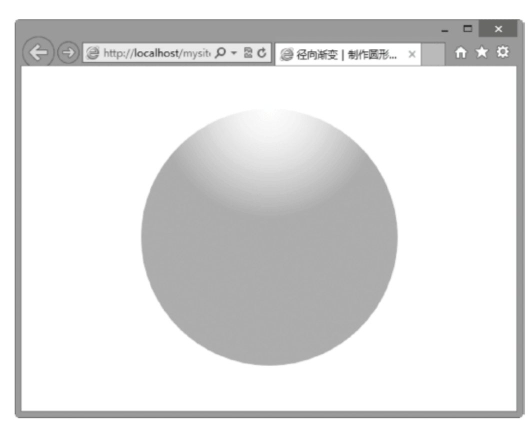


图 4.37 设计发光的球体

本例主要代码如下。

```
<style type="text/css">
* {margin: 0; padding: 0;}
div {
    width: 300px; height: 300px; margin: 50px auto;
    border-radius: 100%;
    background-image: radial-gradient(8em circle at top, hsla(220,89%,100%,1), hsla(30,60%,60%,.9));
}
</style>

<div></div>
```



视频讲解

4.4.6 设计图标

本例通过 CSS3 径向渐变制作圆形图标按钮，用到的知识点主要包括：使用 radial-gradient 属性定义网页背景，以及按钮被激活状态的径向渐变效果；使用 background-image 属性定义多重背景效果，其中一个为浅灰色亮面，另一个是深陷的暗点；使用 background-position 属性把这两个绘制的背景图像叠加在一起；使用 background-size 属性定义多重背景显示大小为 16px*16px，然后按默认状态平铺显示。设计效果如图 4.38 所示。





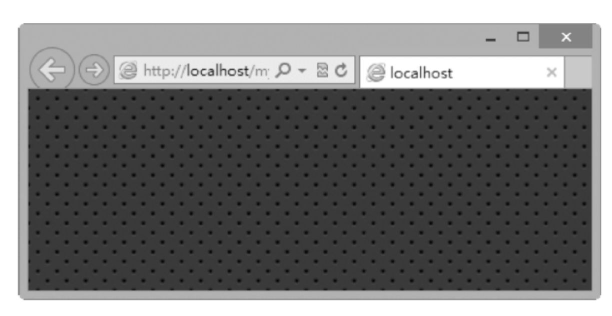


图 4.38　定义网页麻点背景效果

使用@font-face 命令导入外部字体 font/icomoon.eot，定义字体图形效果。

使用 radial-gradient 属性为按钮标签定义径向渐变，设计立体按钮效果，使用 border-radius: 50%; 声明定义按钮圆形显示，使用 box-shadow 属性为按钮添加投影效果。

使用 text-shadow 属性为按钮文本定义阴影效果，当鼠标经过按钮时，使用 text-shadow 属性设计文本发亮显示。

当按钮被激活时，使用 box-shadow 属性定义按钮内阴影，增亮按钮效果，使用 radial-gradient 设计环形径向渐变效果，为按钮添加晕边效果。

示例效果如图 4.39 所示。

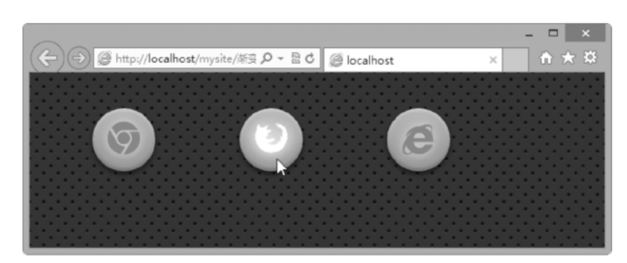


图 4.39　设计径向渐变图标按钮效果

主要页面代码如下。

```
<style type="text/css">
body {
    background-color: #282828;
    background-image: radial-gradient(black 15%, transparent 16%), radial-gradient(black 15%, transparent 16%), radial-gradient(rgba(255, 255, 255, 0.1) 15%, transparent 20%), radial-gradient(rgba(255, 255, 255, 0.1) 15%, transparent 20%);
    background-position: 0 0px, 8px 8px, 0 1px, 8px 9px;
    background-size: 16px 16px;
}
@font-face {
    font-family: 'icomoon';
    src: url('font/icomoon.eot');
```





```
        src: url('font/icomoon.eot?#iefix') format('embedded-opentype'), url('font/icomoon.svg#icomoon') format('svg'),
url('font/icomoon.woff') format('woff'), url('font/icomoon.ttf') format('truetype');
        font-weight: normal;
        font-style: normal;
        }
    .controls_button {width: 500px; margin: 40px auto;}
    .button {
        width: 70px; height: 70px; margin-right: 90px;
        font-size: 0; border: none;
        border-radius: 50%;
        box-shadow: 0 1px 5px rgba(255,255,255,.5) inset, 0 -2px 5px rgba(0,0,0,.3) inset, 0 3px 8px rgba(0,0,0,.8);
        background: radial-gradient(circle at top center, #f28fb8, #e982ad, #ec568c);
        }
    .button:nth-child(3) { margin-right: 0; }
    .button:after {
        font-family: 'icomoon';
        speak: none;
        font-weight: normal;
        -webkit-font-smoothing: antialiased;
        font-size: 36px;
        content: "\21";
        color: #dd5183;
        text-shadow: 0 3px 10px #f1a2c1, 0 -3px 10px #f1a2c1;
        }
    .button:nth-child(2):after { content: "\22"; }
    .button:nth-child(3):after { content: "\23"; }
    .button:hover:after { color: #fff; text-shadow: 0 1px 20px #fccdda, 1px 0 14px #fccdda;}
    .button:active {
        box-shadow: 0 2px 7px rgba(0,0,0,.5) inset, 0 -3px 10px rgba(0,0,0,.1) inset, 0 1px 3px rgba(255,255,255,.5);
        background: radial-gradient(circle at top center, #f28fb8, #e982ad, #ec568c);
        }
    </style>

    <div class="controls_button">
        <button type="button" class="button">Chrome</button>
        <button type="button" class="button">Firefox</button>
        <button type="button" class="button">IE</button>
    </div>
```

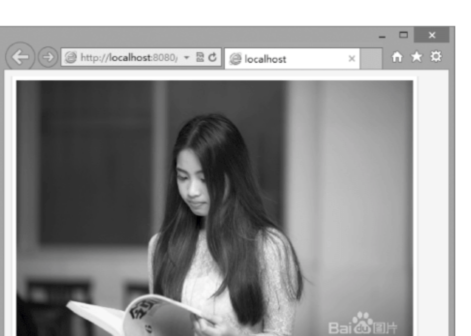






4.4.7　设计图片水印

本例利用 opacity 属性设计水印特效，同时利用 CSS 定位技术实现水印与图片重叠显示，演示效果如图 4.40 所示。



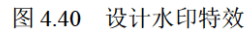


图 4.40　设计水印特效

【操作步骤】

第 1 步，启动 Dreamweaver，新建一个网页，保存为 test.html。

第 2 步，构建一个简单的 HTML 结构。设计一个包含框（<div class="watermark">），主要是为水印图片能够在照片上精确定位提供一个参照，并把它们都捆绑在一起，避免在网页中随处浮动。插入的第 1 幅图片为照片，第 2 幅图片为水印图片，代码如下。

```
<div class="watermark">
    <img src="images/bg.jpg" class="img" width="400">
    <img src="images/logo.png" class="logo" width="100">
</div>
```

在没有任何样式的情况下，显示如图 4.41 所示的效果。

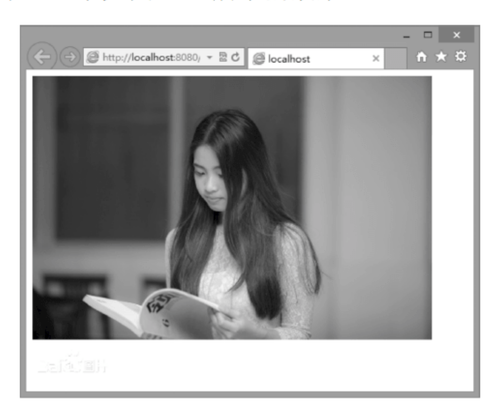


图 4.41　插入图像后的效果

第 3 步，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后准备输入样式。

第 4 步，首先定义包含元素为相对定位，这样能够保证水印图片定位在照片上面。由于 div 元素默认显示块状态元素，宽度为 100%，此时无法精确定位内部的水印。如果固定宽度，就使设计失去了灵活性，还需要确定包含元素和照片的宽度。具体样式如下。

```
.watermark {                    /* 包含块样式 */
    position:relative;          /* 相对定位 */
    float:left;                 /* 向左浮动，这样包含元素能够自动包裹包含的照片 */
    display:inline;             /* 行内显示，这样就避免包含元素随处浮动 */
}
```

第 5 步，下面定义一个类样式，设计阴影图片效果。

```
.img {
    background: white;
    padding: 5px 5px 9px 5px;
    background: white url(images/shad_bottom.gif) repeat-x bottom left;
    border-left: 2px solid #dcd7c8;
    border-right: 2px solid #dcd7c8;
}
```

第 6 步，定义水印透明度和精确定位的位置，具体样式如下。

```
.img1 {
    filter:alpha(opacity=40);       /* 兼容 IE 浏览器 */
    -moz-opacity:0.4;               /* 兼容 Moz 和 FF 浏览器 */
    opacity: 0.4;                   /* 支持 CSS3 的浏览器（FF 1.5 也支持）*/
    position:absolute;              /* 绝对定位 */
    right:20px;                     /* 定位到照片的右侧 */
    bottom:20px;                    /* 定位到照片的底部 */
}
```

IE 使用专有的 CSS 滤镜 filter:alpha(opacity)来定义对象的透明度，而 Moz Family 浏览器使用私有属性-moz-opacity 来定义，对于标准浏览器来说，一般都支持标准属性 opacity。上面的样式定义水印图片的透明度为 0.4，位于照片的右下角。

4.5 在线练习


在线练习

使用 CSS3 设计各种网页图像效果，以及各种网页背景图像特效。感兴趣的读者可以扫码练习。



使用 CSS 美化超链接

在网页中，超链接是最常用的对象，网站的所有页面都由超链接串接而成，超链接完成了页面之间的跳转。此外，超链接也是目前浏览者与服务器交互的主要手段之一。当单击包含超链接的文字、图片或其他对象时，浏览器会根据其指示载入一个新的页面，或者跳转到指定位置，或者执行特定任务。通过 CSS，可以设置超链接和导航栏的样式及其外观属性。

【学习要点】

- 认识伪类和伪对象。
- 定义超链接样式。
- 能够灵活设计符合页面风格的链接样式。





Note



视频讲解

5.1　超链接基本样式

在网页中，被超链接包裹的字体颜色默认为蓝色，链接文本默认显示一条下划线，当鼠标指针移到超链接对象上时，鼠标指针变成手形。如果超链接被访问，那么链接文本颜色就会发生改变，显示为紫色。不过用户可以根据网站或页面设计风格使用 CSS 重新定义超链接的样式。

使用类型选择器 a 可以设置超链接的默认样式。例如，下面的样式可以将所有链接文本设置为红色。

```
a{ color: red;}
```

超链接存在 4 种状态，同时直接为<a>标签定义样式，会影响锚点的样式。一般情况下锚点是不要显示出来的，为了避免这个问题，CSS 为 a 元素提供了如下 4 个状态伪类选择器定义链接样式。

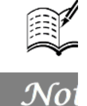
- ☑ a:link：链接默认的样式。
- ☑ a:visited：链接已被访问过的样式。
- ☑ a:hover：鼠标指针在链接上的样式。
- ☑ a:active：单击链接时的样式。

【示例 1】在本示例中定义页面所有超链接默认为红色下划线效果，当鼠标指针经过时显示为绿色下划线效果，而当单击超链接时则显示为黄色下划线效果，超链接被访问过之后显示为蓝色下划线效果，代码如下，演示效果如图 5.1 所示。

```
<style type="text/css">
a:link {color: #FF0000;/* 红色 */}              /* 超链接默认样式 */
a:visited {color: #0000FF; /* 蓝色 */}          /* 超链接被访问后的样式 */
a:hover {color: #00FF00; /* 绿色 */}            /* 鼠标指针经过超链接的样式 */
a:active {color: #FFFF00; /* 黄色 */}           /* 超链接被激活时的样式 */
</style>

<ul class="p1">
    <li><a href="#" class="a1">首页</a></li>
    <li><a href="#" class="a2">新闻</a></li>
    <li><a href="#" class="a3">微博</a></li>
</ul>
<ul class="p2">
    <li><a href="#" class="a1">关于</a></li>
    <li><a href="#" class="a2">版权</a></li>
    <li><a href="#" class="a3">友情链接</a></li>
</ul>
```





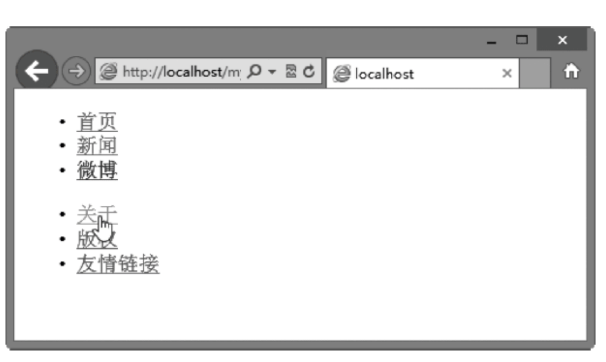


图 5.1　定义超链接样式

【示例 2】针对示例 1 的文档结构，如果要定义第一个列表内超链接样式，则可以使用包含选择器来定义，代码如下。

```
a:link {color: #FF0000;/* 红色 */}          /* 超链接默认样式 */
a:visited {color: #0000FF; /* 蓝色 */}      /* 超链接被访问后的样式 */
a:hover {color: #00FF00; /* 绿色 */}        /* 鼠标指针经过超链接的样式 */
a:active {color: #FFFF00; /* 黄色 */}       /* 超链接被激活时的样式 */
.p1 a:link {color: #FF0000;}
.p1 a:visited {color: #0000FF;}
.p1 a:hover {color: #00FF00;}
.p1 a:active {color: #FFFF00;}
```

【示例 3】用户可以使用限定选择器定义超链接样式。所谓限定选择器，就是在伪类、类或 ID 选择器前面附加一个子选择器，限定后面子选择器所匹配的范围。注意，限定选择器中间没有空格，代码如下。

```
.a1:link {color: #FF0000;}
.a1:visited {color: #0000FF; }
.a1:hover {color: #00FF00;}
.a1:active {color: #FFFF00;}
```

也可以双重限定，代码如下。

```
a.a1:link {color: #FF0000;}
a.a1:visited {color: #0000FF; }
a.a1:hover {color: #00FF00;}
a.a1:active {color: #FFFF00;}
```

注意：在定义超链接样式时，超链接的 4 种状态顺序是有要求的，一般不能随意调换。正确顺序应该是：link→visited→hover→active。





【示例 4】在下面样式中，当鼠标指针经过超链接时，会先执行第 1 行声明，但是紧接着第 3 行声明的样式会覆盖第 1 行和第 2 行声明的样式，所以就无法看到鼠标指针经过和被激活时的效果。

```
a.a1:hover {color: #00FF00;}
a.a1:active {color: #FFFF00;}
a.a1:link {color: #FF0000;}
a.a1:visited {color: #0000FF; }
```

【示例 5】超链接的 4 种状态并非都要定义，可以根据需要定义其中的两个或 3 个。如果要把未访问的和已经访问的链接定义成相同的样式，则可以定义 link、hover 和 active 3 种状态，代码如下。

```
a.a1:link {color: #FF0000;}
a.a1:hover {color: #00FF00;}
a.a1:active {color: #FFFF00;}
```

【示例 6】如果仅希望超链接显示两种状态样式，可以使用 a 和 hover 来定义。其中<a>标签选择器定义 a 元素的默认显示样式，然后定义鼠标指针经过时的样式，代码如下。

```
a {color: #FF0000;}
a:hover {color: #00FF00;}
```

【示例 7】如果页面中还包括锚记，将会影响锚记的样式。如果定义如下的样式，则仅影响超链接未访问时的样式和鼠标指针经过时的样式，代码如下。

```
a:link {color: #FF0000;}
a:hover {color: #00FF00;}
```

5.2 案例实战

超链接对象可以显示为多种样式，如动画、按钮、图像、特效等，本节将通过多个案例介绍常用链接样式的设计技巧。



视频讲解

5.2.1 设计下划线样式

很多用户不喜欢超链接的默认下划线效果，那么可以使用下面的样式清除它。

```
a {/* 完全清除超链接的下划线效果 */
    text-decoration:none;
}
```

然后设计在鼠标指针经过时增加下划线，因为下划线具有很好的提示作用，代码如下。

```
a:hover {/* 鼠标指针经过时显示下划线效果 */
```





```
        text-decoration:underline;
    }
```

下划线样式不仅仅是一条实线，可以根据需要自定义设计。主要设计思路如下。

☑　借助<a>标签的底边框线来实现。

☑　利用背景图像来实现。利用背景图像可以设计出更多精巧的下划线样式。

【示例 1】本示例设计当鼠标指针经过时显示虚下划线、加粗、加重色彩的效果，把这个样式表引入到页面中，则超链接的显示效果如图 5.2 所示。

```
<style type="text/css">
a {/* 超链接的默认样式 */
    text-decoration:none;              /* 清除超链接下划线 */
    color:#999;                        /* 浅灰色文字效果 */
    }
a:hover {/*鼠标指针经过时样式 */
    border-bottom:dashed 1px red;      /* 鼠标指针经过时显示虚下划线效果 */
    color:#000;                        /* 加重颜色显示 */
    font-weight:bold;                  /* 加粗字体显示 */
    zoom:1;                            /* 解决 IE 浏览器无法显示问题 */
    }
</style>

<p>下载<a href="#">HTML 文档</a> 和 <a  href="#">CSS 文件</a></p>
```



图 5.2　定义下划线样式

【示例 2】在本示例中，定义超链接始终显示为下划线效果，并通过颜色变化来提示鼠标指针经过时的状态变化。代码如下，效果如图 5.3 所示。

```
<style type="text/css">
a {/* 超链接的默认样式 */
    text-decoration:none;              /* 清除超链接下划线 */
    border-bottom:dashed 1px red;      /* 红色虚下划线效果 */
    color:#666;                        /* 灰色字体效果 */
    zoom:1;                            /* 解决 IE 浏览器无法显示的问题 */
```





```
        }
    a:hover {/* 鼠标指针经过时样式 */
        color:#000;                            /* 加重颜色显示 */
        border-bottom:dashed 1px #000;         /* 改变虚下划线的颜色 */
        }
    </style>

    <p>下载<a href="#">HTML 文档</a> 和 <a   href="#">CSS 文件</a></p>
```



图 5.3　定义下划线颜色

【示例 3】由于浏览器在解析虚线时的效果并不一致，且显示效果不是很精致，如果使用背景图来定义下划线的虚线样式，则效果会更好。

第 1 步，使用 Photoshop 设计一个虚线图，如图 5.4 所示是一个放大 32 倍的虚线段设计图效果（images/dashed.psd），在设计时应该确保高度为 1 像素，宽度可以为 4 像素、6 像素或 8 像素。这个主要根据虚线的疏密进行设置，然后选择一种颜色以跳格方式进行填充，最后保存为 GIF 格式图像即可。当然最佳视觉空隙是间隔两个像素空格。



图 5.4　使用 Photoshop 设计虚线段

第 2 步，修改示例 2 中的下划线样式，使用背景图代替 border-bottom:dashed 1px red;声明，主要样式代码如下，预览效果如图 5.5 所示。

```
    a {/* 超链接的默认样式 */
        text-decoration:none;                  /* 清除超链接下划线 */
        color:#666;                            /* 灰色字体效果 */
        }
    a:hover {/* 鼠标指针经过时样式 */
```



```
        color:#000;                                             /* 加重颜色显示 */
        /* 定义背景图像，定位到超链接元素的底部，并沿 x 轴水平平铺 */
        background:url(images/dashed1.gif) left bottom repeat-x;
    }
```





图 5.5　背景图像设计的下划线样式

提示： 有关下划线的效果还有很多，只要巧妙结合超链接的底部边框、下划线和背景图像，就可以设计出很多有个性的样式。例如，定义下划线的颜色、下划线距离、下划线长度和对齐方式以及定制双下划线等。

5.2.2　设计动态下划线样式



视频讲

如果使用 GIF 动画作为背景图像，来模拟超链接的下划线样式，则可以让下划线动起来，演示效果如图 5.6 所示。



图 5.6　设计动态下划线样式

示例代码如下。

```
<style type="text/css">
a, a:visited {
    background: url(images/bg2.gif) repeat-x left bottom;
    text-decoration: none;
}
a:hover {
    background: url(images/bg.gif) repeat-x left bottom;
    text-decoration: none;
}
</style>

<p>下载<a href="#">HTML 文档</a> 和 <a href="#">CSS 文件</a></p>
```



5.2.3 设计按钮样式



Note



视频讲解

设计立体按钮效果的一般方法如下。

第一，利用边框线的颜色变化来制造视觉错觉。可以把右边框和底部边框相结合，把顶部边框和左边框相结合，利用明暗色彩的搭配来设计立体变化效果。

第二，利用超链接背景色的变化来营造凸凹变化的效果。超链接的背景色可以设置相对较深的颜色，以营造凸起效果，当鼠标指针经过时，再定义浅色背景来营造凹下效果。

第三，利用环境色、字体颜色（前景色）来烘托立体变化过程。

【示例 1】在本示例中，定义超链接在默认状态下显示灰色右边框线、灰色底边框线效果。而当鼠标指针经过时，则清除右侧和底部边框线，并定义左侧和顶部边框效果，这样利用错觉设计出一个简单的凸凹立体效果。代码如下，演示效果如图 5.7 所示。

```
<style type="text/css">
a {/* 超链接的默认样式 */
    text-decoration:none;                    /* 清除超链接下划线 */
    border-right:solid 1px #666;             /* 灰色右边框线 */
    border-bottom:solid 1px #666;            /* 灰色底边框线 */
    zoom:1;                                  /* 解决 IE 浏览器无法显示问题*/
    }
a:hover {/* 鼠标经过时样式 */
    border-left:solid 1px #666;              /* 灰色左边框线 */
    border-top:solid 1px #666;               /* 灰色顶边框线 */
    border-right:none;                       /* 清除右边框线 */
    border-bottom:none;                      /* 清除底边框线 */
    }
</style>

<p>下载<a href="#">HTML 文档</a> 和 <a  href="#">CSS 文件</a></p>
```

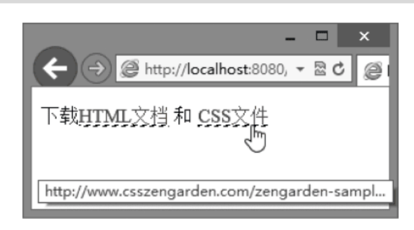


图 5.7 定义简单的立体样式

示例 1 的立体效果不是很明显，但是如果结合前景色和背景色的变化，以及页面背景色的衬托，就可以设计更立体化的超链接效果。

【示例 2】在本示例中，结合网页背景色、超链接的背景色和前景色，设计一个更富有立体效果的超链接样式。代码如下，演示效果如图 5.8 所示。





```
<style type="text/css">
body { background:#fcc; /* 浅色背景 */}            /* 网页背景颜色 */
a {/* 超链接的默认样式 */
    text-decoration:none;                           /* 清除超链接下划线 */
    border:solid 1px;                               /* 定义 1 像素实线边框 */
    padding: 0.4em 0.8em;                           /* 增加超链接补白 */
    color: #444;                                    /* 定义灰色字体 */
    background: #f99;                               /* 超链接背景色 */
    border-color: #fff #aaab9c #aaab9c #fff;        /* 分配边框颜色 */
    zoom:1;                                         /* 解决 IE 浏览器无法显示的问题*/
    }
a:hover {/* 鼠标指针经过时样式 */
    color: #800000;                                 /* 超链接字体颜色 */
    background: transparent;                        /* 清除超链接背景色 */
    border-color: #aaab9c #fff #fff #aaab9c;        /* 分配边框颜色 */
    }
</style>

<p>下载<a href="#">HTML 文档</a> 和 <a   href="#">CSS 文件</a></p>
```

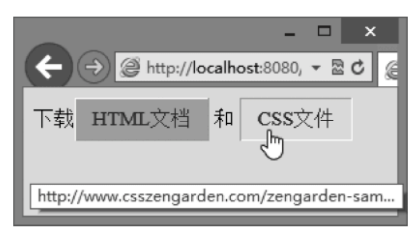


图 5.8　定义逼真的立体样式

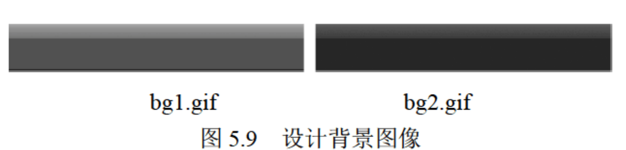
5.2.4　设计背景图像交换样式



背景图像交换样式的设计技巧：利用相同大小但不同效果的背景图像进行轮换，模拟复杂的鼠标动态效果。因此，图像样式的关键是背景图像的设计和几种不同效果的背景图像是否能够过渡自然、切换吻合。

【示例 1】下面通过一个案例进行演示说明。

第 1 步，使用 Photoshop 设计两幅大小相同但效果略不同的图像，如图 5.9 所示。图像的大小为 200px*32px，第 1 张图像设计风格为渐变灰色，并带有玻璃效果，第 2 张图像设计风格为深黑色渐变。



bg1.gif　　bg2.gif

图 5.9　设计背景图像





第 2 步，在 Dreamweaver 中使用两张背景图像设计超链接的样式。页面完整代码如下。

```
<style type="text/css">
a {/* 超链接的默认样式 */
    text-decoration:none;                          /* 清除默认的下划线 */
    display:inline-block;                          /* 行内块状显示 */
    padding:2px 1em;                               /* 为文本添加补白效果 */
    height:28px;                                   /* 固定高度 */
    line-height:32px;                              /* 行高等于高度，设计垂直居中 */
    text-align:center;                             /* 文本水平居中 */
    background:url(images/b1.gif) no-repeat center;  /* 定义背景图像 1，禁止平铺，居中 */
    color:#ccc;                                    /* 浅灰色字体 */
    }
a:hover {/* 鼠标指针经过时样式 */
    background:url(images/b2.gif) no-repeat center;  /* 定义背景图像 2，禁止平铺，居中 */
    color:#fff;                                    /* 白色字体 */
    }
</style>

<p>下载<a href="#">HTML 文档</a> 和 <a  href="#">CSS 文件</a></p>
```

在上面样式代码中，先定义超链接以行内块状显示，这样便于控制它的宽和高，然后根据背景图像大小定义 a 元素的大小，并分别为默认状态和鼠标指针经过状态下定义背景图像。

超链接的宽度可以不必等于背景图像的宽度，只要小于背景图像的宽度即可。但是高度必须保持与背景图像的高度一致。在设计中可以结合背景图像的效果定义字体颜色。

第 3 步，在浏览器中预览，所得到的超链接效果如图 5.10 所示。



图 5.10　背景图像交换链接效果

【示例 2】为了减少两幅背景图像的 HTTP 请求次数，避免占用不必要的带宽，可以把交换的两幅图像合并为一幅图像，然后利用 CSS 定位技术控制背景图像显示区域。

例如，对于示例 1 中的背景图像，可以将其合并为一幅图像，如图 5.11 所示。



图 5.11　合并背景图像



然后，在超链接中直接引用该合成图像，合成背景图像高度增加为 64px，在默认状态和鼠标指针经过时仅部分背景可见，设计的 CSS 代码如下。

```
a {/* 超链接的默认样式 */
    text-decoration:none;                                   /* 清除默认的下划线 */
    display:inline-block;                                   /* 行内块状显示 */
    padding:2px 1em;                                        /* 为文本添加补白效果 */
    height:28px;                                            /* 固定高度 */
    line-height:32px;                                       /* 行高等于高度，设计垂直居中 */
    text-align:center;                                      /* 文本水平居中 */
    background:url(images/b3.gif) no-repeat center top;     /* 定义背景图像 1，禁止平铺，居中 */
    color:#ccc;                                             /* 浅灰色字体 */
    }
a:hover {/* 鼠标指针经过时样式 */
    background-position:center bottom;                      /* 定位背景图像，显示下半部分 */
    color:#fff;                                             /* 白色字体 */
    }
```

本例与上例设计思路基本相同，主要是在引用外部图像时所有背景图像组合在一张图中，然后利用 CSS 精确定位，显示一半图像，以实现在不同状态下显示不同的背景图像。

使用背景图像设计超链接样式比较实用，且所设计的效果可以模拟各种效果，只要背景图设计新颖、漂亮即可。

5.2.5　设计鼠标指针样式



在默认状态下，鼠标指针经过超链接时显示为手形，使用 CSS 的 cursor 属性可以改变这种默认效果。cursor 属性定义鼠标指针在指定对象上的样式，取值说明如表 5.1 所示。

表 5.1　cursor 属性取值说明

值	说　明
auto	基于上下文决定应该显示什么光标
crosshair	十字线光标（+）
default	基于平台的默认光标。通常渲染为一个箭头
pointer	指针光标，表示一个超链接
move	十字箭头光标，用于标示对象可被移动
e-resize、ne-resize、nw-resize、n-resize、se-resize、sw-resize、s-resize、w-resize	表示正在移动某个边，如 se-resize 光标用来表示框的移动开始于东南角
text	表示可以选择文本。通常渲染为 I 形光标





续表

值	说　明
wait	表示程序正忙，需要用户等待，通常渲染为手表或沙漏
help	光标下的对象包含帮助内容，通常渲染为一个问号或一个气球
<uri>URL	自定义光标类型的图标路径

表 5.1 所列的是 W3C 推荐的标准光标类型，但是 IE 也自定义了不少专有属性值。例如，对于手形光标类型，IE 提供了 hand 专有属性，而标准属性值为 pointer，如果考虑兼容问题，可以按如下方法设计，保证在不同浏览器中都显示为手形光标。

```
a {
    cursor:pointer;                     /* 鼠标经过时手形样式 */
    cursor:hand;                        /* 兼容 IE6 以下版本浏览器 */
    }
```

如果自定义光标样式，使用绝对或相对 URL 指定光标文件（后缀为.cur 或者.ani）。

【示例】本示例定义了鼠标指针样式为十字靶心。代码如下，演示效果如图 5.12 所示。

```
<style type="text/css">
a {/* 超链接的默认样式 */
    text-decoration:none; display:inline-block; padding:2px 1em; height:28px; line-height:32px; text-align:center;
background:url(images/b1.gif) no-repeat center; color:#ccc; }
a:hover {/* 鼠标经过时样式 */
    background:url(images/b2.gif) no-repeat center; color:#fff;
    cursor:url('images/Cursor_3.cur'), url('images/Cursor_3.gif'); /* 自定义光标样式 */
}
</style>

<p>下载<a href="#">HTML 文档</a> 和 <a  href="#">CSS 文件</a></p>
```



图 5.12　自定义光标样式效果

【补充】

使用自定义图像作为光标类型，IE 和 Opera 只支持*.cur 等特定的图片格式；而 Firefox、Chrome 和 Safari 既支持特定图片类型，也支持常见的*.jpg、*.gif、*.jpg 等图片格式。







cursor 属性值可以是一个序列，当用户端无法处理第 1 个图标时，它会尝试处理第 2 个、第 3 个等，如果用户端无法处理任何定义的光标，它必须使用列表最后的通用光标。例如，下面样式中就定义了 3 个自定义动画光标文件，最后定义了一个通用光标类型。

```
a:hover { cursor:url('images/1.ani'), url('images/1. cur'), url('images/1.gif'), pointer;}
```

5.2.6　设计图片按钮样式

在网页中会经常看见使用图像设计的超链接按钮。为超链接设计图像样式可以有多种方法，其中最常用的方法是借助 CSS 的 background-image 属性来定义超链接的背景图像。

【示例 1】本示例通过用背景图像替换超链接文本，来设计与页面风格相统一的超链接效果，演示效果如图 5.13 所示。

```
<style type="text/css">
a.reg { /*  超链接样式  */
    background: transparent url('images/btn2.gif') no-repeat top left; /*  背景图像  */
    display: block;              /*  块状显示，方便定义宽度和高度  */
    width:74px;                  /*  宽度，与背景图像同宽  */
    height: 25px;                /*  高度，与背景图像同高  */
    text-indent:-999px;          /*  隐藏超链接中的文本  */
}
</style>

<a class="reg"   href="#">注册</a>
```



图 5.13　使用图形化按钮设计超链接样式

注意：在使用背景图像来设计超链接时应注意以下几个问题。

- 如果完全使用背景图像来设计一个超链接样式，应防止背景图像的重复平铺，可以使用 background-repeat 属性禁止平铺。
- 定义<a>标签以块状或者行内块状显示，以方便为超链接定义高和宽。在定义超链接的显示大小时，其宽和高最好与背景图像保持一致。也可以使用 padding 属性撑开<a>标签，以代替 width 和 height 属性声明。
- 应使用 text-indent 属性隐藏超链接中的文本。
- 如果超链接的区域比背景图像大，应使用 background-position 属性定位背景图像在超链接中的显示位置。





Note

【示例 2】使用图像也能够设计出富有动态的超链接按钮。例如，本示例为超链接不同状态定义不同背景图像：当在正常状态下，超链接左侧显示一个箭头式的背景图像；当鼠标指针经过超链接时，背景图像被替换为另一个动态 GIF 图像，使整个超链接动态效果立即显示出来。代码如下，演示效果如图 5.14 所示。

```
<style type="text/css">
a.reg {/*  定义超链接正常样式：定位左侧背景图像  */
    background: url("images/arrow2.gif") no-repeat left center;
    padding-left:14px;
    }
a.reg:hover {/*  定义鼠标指针经过时超链接样式：定位左侧背景图像  */
    background: url("images/arrow1.gif") no-repeat left center;
    padding-left:14px;
    }
</style>

<a class="reg"   href="#">注册</a>
```

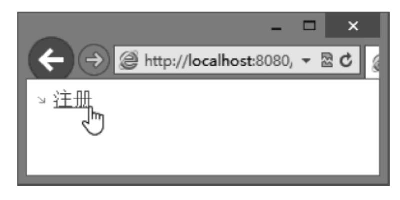


图 5.14　动态背景图像超链接样式

在上面的样式表中，通过 padding-left 属性定义超链接左侧空隙，这样就可以使定义的背景图像显示出来，避免被链接文本所遮盖。实战中，经常需要使用 padding 属性来为超链接增加空余的空间，以便背景图像能够很好地显示出来。

在设计时，巧妙地结合超链接的几种不同显示状态，为不同状态设计不同的背景图像，用户会看到很多有趣的效果，而且可以设计出更富有提示性的超链接效果。例如，在超链接左侧定义一个箭头式的背景图像，当鼠标指针经过时再定义箭头图像显示在超链接的右侧，这样就会给人一种错觉，即当鼠标指针经过时，超链接如同离弦的箭头。或者设计当超链接被单击之后，在超链接的左侧或者右侧显示一个对勾标记，提示该超链接已经被访问过。



视频讲解

5.2.7　设计滑动背景样式

在 CSS 中，一个经常被讨论的设计技巧就是背景图的可层叠性，并允许在彼此之上进行滑动，以创造一些特殊的效果，这就是滑动门特效。

很多用户喜欢把一个绘制好的图像分成两截，让其中的一截固定在超链接按钮的一端，另一截固定在另一端，如果背景图足够大的话，会发现不管超链接包含的字数有多少，字体有多大，它都能够很好地适应伸缩的按钮，使设计效果总是显示为一致。





图 5.15 直观地展示了可扩展的图形按钮的设计原理。为了帮助读者更直观地理解和体会，下面结合一个示例进行具体讲解。

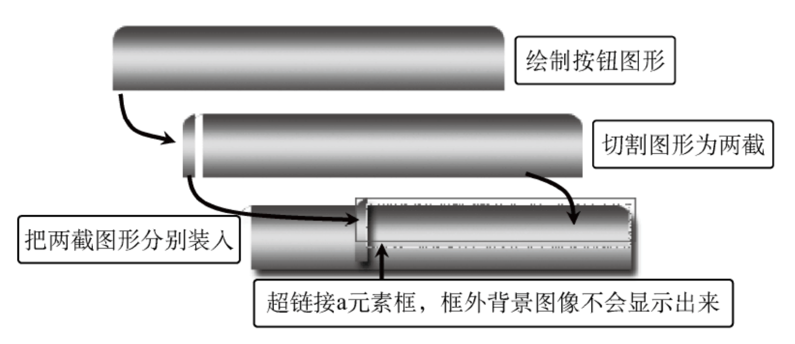


图 5.15 滑动门设计原理示意图

【操作步骤】

第 1 步，使用 Photoshop 设计好按钮图形的效果图，然后分切为两截，其中一截应尽可能窄，只包括一条椭圆边，另一截可以尽可能大，这样设计的图形按钮就可以容纳更多的字符，如图 5.16 所示。



图 5.16 绘制并裁切滑动门背景图

第 2 步，启动 Dreamweaver，新建网页，保存为 test.html，在<body>标签内输入以下代码。构建一个可以定义重叠背景图的超链接结构，具体结构如下，在每个超链接<a>标签中包含了一个<span>辅助标签。

```
<a  href="#"><span>按钮</span></a>
<a  href="#"><span>超链接</span></a>
<a  href="#"><span>图像按钮</span></a>
<a  href="#"><span>扩展性按钮</span></a>
<a  href="#"><span>能够定义很多字数的文本链接</span></a>
```

第 3 步，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后输入下面的样式。使用 CSS 把短的背景图（left1.gif）固定在<a>标签的左侧。

```
a {/* 定义超链接样式*/
    background: url('images/left1.gif') no-repeat top left; /* 把短截背景图像固定在左侧*/
    display: block;                /* 以块状显示，这样能够定义大小 */
    float:left;                    /* 浮动显示，这样 a 元素能够自动收缩宽度，以正好包容文本 */
    padding-left: 8px;             /* 增加左侧内边距，该宽度正好与上面定义的背景图像同宽 */
    font: bold 13px Arial;         /* 超链接文本字体属性 */
    line-height: 22px;             /* 定义行高 */
    height: 30px;                  /* 定义按钮高度 */
```

Note

```
    color: white;                          /* 字体颜色 */
    margin-left:6px;                       /* 左侧外边框 */
    text-decoration:none;                  /* 清除默认的下划线样式 */
    }
```

第 4 步，把长的背景图（right1.gif）固定在<span>标签的右侧。

```
a span {
    background: url('images/right1.gif') no-repeat top right;    /* 定义长截背景图像 */
    display: block;                                              /* 块状显示 */
    padding: 4px 10px 4px 2px;                                   /* 增加内边距 */
    }
```

第 5 步，在浏览器中预览，显示效果如图 5.17 所示。如果希望当鼠标指针经过时让超链接样式稍稍有点变化，以增加按钮的动态感，不妨给鼠标经过时增加一个下划线效果，代码如下。

```
a:hover { text-decoration: underline;}
```



图 5.17　设计滑动门链接效果



视频讲解

5.2.8　设计超链接类型样式

在下面的示例中，将模拟百度文库的“相关文档推荐”模块样式设计效果，演示如何利用属性选择器快速并准确匹配文档类型，为不同类型的文档超链接定义不同的显示图标，以便浏览者准确识别文档类型。示例演示效果如图 5.18 所示。

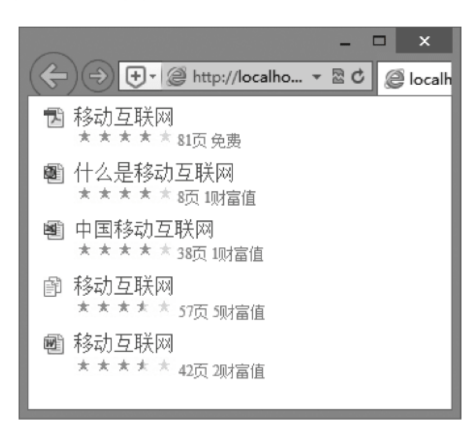


图 5.18　设计超链接文档类型的显示图标





【操作步骤】

第 1 步，构建一个简单的模块结构。在这个模块结构中，为了能够突出重点，忽略了其他细节信息，代码如下。

```
<div id="wrap">
    <p><a href="http://www.baidu.com/name.pdf">移动互联网</a><span><img src="images/star1.jpg" /> 81 页 免费</span> </p>
    <p><a href="http://www.baidu.com/name.ppt">什么是移动互联网</a><span><img src="images/star1.jpg" /> 8 页 1 财富值</span> </p>
    <p><a href="http://www.baidu.com/name.xls">中国移动互联网</a><span><img src="images/star1.jpg" /> 38 页 1 财富值 </span> </p>
    <p><a href="http://www.baidu.com/name.txt">移动互联网</a> <span><img src="images/star3.jpg" /> 57 页 5 财富值</span></p>
    <p><a href="http://www.baidu.com/name.doc">移动互联网</a><span><img src="images/star3.jpg" /> 42 页 2 财富值</span> </p>
</div>
```

第 2 步，新建一个内部样式表，在样式表中对案例文档进行样式初始化，代码如下。

```
/*初始化超链接、sapn 元素和 p 元素基本样式*/
a { padding-left: 24px; text-decoration: none; }
span { color: #999; font-size: 12px; display: block; padding-left: 24px; padding-bottom: 6px; }
p { margin: 4px; }
```

第 3 步，利用属性选择器为不同类型的文档超链接定义显示图标。

```
a[href  $="pdf"] { /*匹配 PDF 文件*/
    background: url(images/pdf.jpg) no-repeat left center;
}
a[href  $="ppt"] { /*匹配演示文稿*/
    background: url(images/ppt.jpg) no-repeat left center;
}
a[href  $="txt"] { /*匹配记事本文件*/
    background: url(images/txt.jpg) no-repeat left center;
}
a[href  $="doc"] { /*匹配 Word 文件*/
    background: url(images/doc.jpg) no-repeat left center;
}
a[href  $="xls"] { /*匹配 Excel 文件*/
    background: url(images/xls.jpg) no-repeat left center;
}
```





【拓展】

超链接的类型和形式是多样的，如锚链接、下载链接、图片链接、空链接、脚本链接等，都可以利用属性选择器来标识这些超链接的不同样式，代码如下。

```
a[href^="http:"] { /*匹配所有有效超链接*/
    background: url(images/window.gif) no-repeat left center;
}
a[href  $="xls"] { /*匹配 XML 样式表文件*/
    background: url(images/icon_xls.gif) no-repeat left center;
    padding-left: 18px;
}
a[href  $="rar"] { /*匹配压缩文件*/
    background: url(images/icon_rar.gif) no-repeat left center;
    padding-left: 18px;
}
a[href  $="gif"] { /*匹配 GIF 图像文件*/
    background: url(images/icon_img.gif) no-repeat left center;
    padding-left: 18px;
}
a[href  $="jpg"] { /*匹配 JPG 图像文件*/
    background: url(images/icon_img.gif) no-repeat left center;
    padding-left: 18px;
}
a[href  $="png"] { /*匹配 PNG 图像文件*/
    background: url(images/icon_img.gif) no-repeat left center;
    padding-left: 18px;
}
```

5.3 在线练习

使用 CSS3 设计各种超链接样式，强化基本功训练。感兴趣的读者可以扫码练习。





第6章 使用 CSS 美化列表

在网页中，大量的信息通过列表结构进行编排，如新闻列表、导航列表、排行列表、分类列表等，可以说列表是管理网页信息最有效的方式之一，使用频率大于标题文本和段落文本。本章将介绍各种网页构建的样式设计，如导航条、菜单栏、新闻栏目、引导页、列表页、图文版式等。

【学习要点】

- 使用 CSS 设计列表基本样式。
- 根据网页具体内容编排列表版式。





Note

6.1 列表基本样式

在默认状态下网页列表呈现的效果是左侧附加项目符号，列表项目缩进显示。CSS 为列表结构定义了 4 个专门属性，说明如表 6.1 所示。

表 6.1　CSS 专用列表属性

属　性	说　明
list-style	复合属性。设置列表项目相关内容
list-style-image	设置列表项目的符号图像
list-style-position	设置列表项目符号的显示位置，根据文本在内或在外排列，取值包括 inside 和 outside
list-style-type	设置列表项目符号的类型

6.1.1　定义项目符号



视频讲解

CSS 使用 list-style-type 属性定义列表项目符号的类型，该属性取值说明如表 6.2 所示。

表 6.2　list-style-type 属性值

属 性 值	说　明	属 性 值	说　明
disc	实心圆，默认值	upper-roman	大写罗马数字
circle	空心圆	lower-alpha	小写英文字母
square	实心方块	upper-alpha	大写英文字母
decimal	阿拉伯数字	none	不使用项目符号
lower-roman	小写罗马数字	armenian	传统的亚美尼亚数字
cjk-ideographic	浅白的表意数字	georgian	传统的乔治数字
lower-greek	基本的希腊小写字母	hebrew	传统的希伯来数字
hiragana	日文平假名字符	hiragana-iroha	日文平假名序号
katakana	日文片假名字符	katakana-iroha	日文片假名序号
lower-latin	小写拉丁字母	upper-latin	大写拉丁字母

CSS 使用 list-style-position 属性定义项目符号的显示位置。该属性取值包括 outside 和 inside，其中 outside 表示把项目符号显示在列表项的文本行以外，列表符号默认显示为 outside，inside 表示把项目符号显示在列表项文本行以内。

【示例】本示例定义项目符号显示为空心圆，位于列表行内部。代码如下，效果如图 6.1 所示。

```
<style type="text/css">
body {/* 清除页边距 */
    margin: 0;                    /* 清除边界 */
```





```
        padding: 0;                      /* 清除补白 */
    }
    ul {/* 列表基本样式 */
        list-style-type: circle;         /* 空心圆符号*/
        list-style-position: inside;     /* 显示在里面 */
        }
    </style>

    <ul>
        <li><a href="#">新闻</a></li>
        <li><a href="#">社区</a></li>
        <li><a href="#">微博</a></li>
        <li><a href="#">微信</a></li>
    </ul>
```



图 6.1　定义列表项目符号

项目符号显示在里面和外面会影响项目符号与列表文本之间的距离，同时影响列表项的缩进效果。不同浏览器在解析时会存在差异。

6.1.2　自定义项目符号



使用 CSS 的 list-style-image 属性可以自定义项目符号。该属性允许指定一个外部图标文件，以此满足个性化设计需求。用法如下。

```
list-style-image：none | <url>
```

默认值为 none。

【示例】以 6.1.1 节示例为基础，增加自定义项目符号。代码如下，效果如图 6.2 所示。

```
body {/* 清除页边距 */
    margin: 0;                           /* 清除边界 */
    padding: 0;                          /* 清除补白 */
}
```




```
ul {/* 列表基本样式 */
    list-style-type:circle;                          /* 空心圆符号*/
    list-style-position:inside;                      /* 显示在里面 */
    list-style-image:url(images/bullet_disk.gif);    /* 自定义列表项目符号 */
}
```



图 6.2　自定义列表项目符号

> 提示：当同时定义项目符号类型和自定义项目符号时，自定义项目符号将覆盖默认的符号类型。但是如果 list-style-type 属性值为 none 或指定外部文件不存在时，则 list-style-type 属性值有效。



视频讲解

6.1.3　使用背景图像设计项目符号

使用背景图定义项目符号的技巧如下。

第一，先隐藏列表结构的默认项目符号。方法是设置 list-style-type:none。

第二，为列表项定义背景图像，指定要显示的项目符号，利用背景图精确定位技术控制其显示位置。同时增加列表项左侧空白，避免背景图被列表文本遮盖。

【示例 1】在本示例中，先清除列表的默认项目符号，然后为项目列表定义背景图像，并定位到左侧垂直居中的位置，为了避免列表文本覆盖背景图像，定义左侧补白为一个字符宽度，这样就可以把列表信息向右缩进显示。代码如下，显示效果如图 6.3 所示。

```
body {/* 清除页边距 */
    margin: 0;                                             /* 清除边界 */
    padding: 0;                                            /* 清除补白 */
}
li {/* 定义列表项目的样式 */
    background-image:url(images/bullet_sarrow.gif);        /* 定义背景图像 */
    background-position:left center;                       /* 精确定位背景图像的位置 */
    background-repeat:no-repeat;                           /* 禁止背景图像平铺显示 */
    padding-left:1em;                                      /* 为背景图像挤出空白区域 */
}
```







图 6.3 使用背景图像设计项目符号

【示例 2】本示例结合超链接的交互状态，设计出富有动态效果的项目符号。代码如下，效果如图 6.4 所示。

```
ul {
    padding: 0;
    margin: 0;
    list-style: none;
    border-bottom: 1px dashed #aaa;
    width:20em;                                          /* 固定项目列表的宽度 */
}
li {padding: 0.5em; border-top: 1px dashed #aaa;}
li a {/* 超链接样式 */
    display:block;                                       /* 块状显示 */
    padding-left:1.5em;                                  /* 为背景图像显示挤出位置 */
    background:url(images/arrow3.gif) left center no-repeat;  /* 固定背景图像在左侧 */
    text-decoration: none;                               /* 清除超链接的下划线 */
}
li a:link {/* 定义未访问超链接背景图像 */
    background:url(images/arrow3.gif) right center no-repeat;  /* 固定背景图像在左侧 */
}
li a:visited {/* 定义已访问超链接背景图像 */
    background:url(images/arrow8.gif) right center no-repeat;  /* 替换左侧背景图像 */
}
li a:hover {/* 定义鼠标指针经过超链接背景图像 */
    background:url(images/arrow4.gif) left center no-repeat;  /* 固定背景图像到右侧 */
}
```





Note

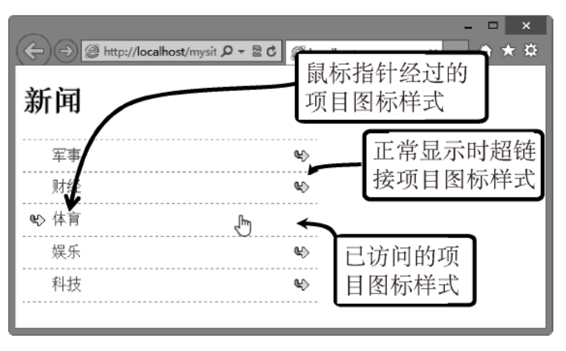


图 6.4 动态图形项目列表符号

6.2 案例实战

下面通过几个案例演示列表样式在栏目设计中的具体应用。



视频讲解

6.2.1 设计堆叠样式

列表结构默认显示为堆叠样式，并以缩进版式进行显示。这是一种符合浏览习惯的布局效果。在新闻列表、分类列表等列表页或栏目中比较常见。

【示例 1】本示例演示了列表结构垂直布局的基本形式。

第 1 步，清除列表结构的默认样式，如列表项目符号、缩进显示格式等。

第 2 步，定义列表项目的宽度和高度，定义其包含的<a>标签以块状显示，设置对应的宽度和高度。

第 3 步，使用背景色、字体颜色和边框颜色等要素设计视觉变化效果，营造鼠标指针经过的动态效果。代码如下，演示效果如图 6.5 所示。

```
<style>
body { background-color: #dee0ff; }
#menu {/* 栏目包含框样式 */
    width: 150px;
    font-family: Arial; font-size: 14px; text-align: right
}
#menu ul {/* 列表外框样式 */
    list-style-type: none;                    /* 不显示项目符号 */
    margin: 0px;                              /* 清除边界 */
    padding: 0px;                             /* 清除补白 */
}
#menu li { border-bottom: 1px solid #9F9FED;}    /* 添加下划线 */
#menu li a {
    display: block;                           /* 块显示 */
```



```
        height: 1em;                              /* 定义行高 */
        padding: 5px 5px 5px 0.5em;               /* 增加内部空隙 */
        text-decoration: none;                    /* 清除下划线 */
        border-left: 12px solid #151571;          /* 左边的粗边 */
        border-right: 1px solid #151571;          /* 右侧阴影 */
}
#menu li a:link, #menu li a:visited {/* 鼠标指针经过时样式 */
        background-color: #1136c1;                /* 背景色 */
        color: #FFFFFF;                           /* 前景色 */
}
#menu li a:hover {      /* 鼠标指针经过时 */
        background-color: #002099;                /* 改变背景色 */
        color: #ffff00;                           /* 改变文字颜色 */
        border-left: 12px solid yellow;           /* 底部增加亮边 */
}
</style>

<div   id="menu">
        <h1>网站导航</h1>
        <ul>
                <li><a href="#" title="">软件工程</a></li>
                <li><a href="#" title="">编程语言</a></li>
                <li><a href="#" title="">软件设计</a></li>
                <li><a href="#" title="">Web 前端</a></li>
                <li><a href="#" title="">手机开发</a></li>
                <li><a href="#" title="">所有随笔</a></li>
        </ul>
</div>
```



图 6.5　列表堆叠样式

6.2.2　设计水平排列样式



第一种，定义列表项目为行内显示，设计所有列表项目在同一行内显示。



Note

【示例1】本示例是在6.2.1节示例基础上，把垂直堆叠形式转换为水平布局形式。

首先，把列表项目定义为行内显示；然后，使用补白（padding）定义列表项目的宽度和高度，因为width和height属性对于行内元素是无效的；最后，利用背景色、边框样式和字体颜色设计超链接的动态效果。代码如下，演示效果如图6.6所示。

```
body { background-color: #dee0ff; }
#menu {font-family: Arial; font-size: 14px; }
#menu ul {/* 列表外框样式 */
    list-style-type: none;                  /* 不显示项目符号 */
    margin: 0px;                            /* 清除边界 */
    padding: 0px;                           /* 清除补白*/
}
#menu li {
    border-bottom: 1px solid #9F9FED; /* 添加下划线 */
    display: inline;                        /* 行内显示 */
}
#menu li a {
    padding: 8px 8px 12px 1em;              /* 通过 padding 撑开列表项目 */
    text-decoration: none;                  /* 清除下划线 */
    border-left: 12px solid #151571;        /* 左边的粗边 */
    border-right: 1px solid #151571;        /* 右侧阴影 */
}
#menu li a:link, #menu li a:visited {/* 已经访问过样式*/
    background-color: #1136c1;
    color: #FFFFFF;
}
#menu li a:hover {      /* 鼠标指针经过时样式 */
    background-color: #002099;              /* 改变背景色 */
    color: #ffff00;                         /* 改变文字颜色 */
    border-left: 12px solid yellow;
}
```



图6.6　水平布局列表样式





第二种，使用浮动显示列表项目，设计水平布局。

【示例 2】本示例将以浮动方式设计导航菜单。代码如下，演示效果如图 6.7 所示。

```
body { background-color: #dee0ff; }
#menu {font-family: Arial; font-size: 14px; }
#menu ul {/* 列表外框样式 */
    list-style-type: none;                    /* 不显示项目符号 */
    margin: 0px;                              /* 清除边界 */
    padding: 0px;                             /* 清除补白*/
}
#menu li {
    border-bottom: 1px solid #9F9FED;         /* 添加下划线 */
    float: left;                              /* 定义列表项目向左浮动显示 */
}
#menu li a {
    width: 120px;                             /* 定义超链接宽度 */
    display: block;                           /* 定义块显示 */
    height: 1em;                              /* 定义行高 */
    padding: 5px 5px 5px 0.5em;               /* 增加内部空隙 */
    text-decoration: none;                    /* 清除下划线 */
    border-left: 12px solid #151571;          /* 左边的粗边 */
    border-right: 1px solid #151571;          /* 右侧阴影 */
}
#menu li a:link, #menu li a:visited {/* 已经访问过样式*/
    background-color: #1136c1;
    color: #FFFFFF;
}
#menu li a:hover {      /* 鼠标指针经过时 */
    background-color: #002099;                /* 改变背景色 */
    color: #ffff00;                           /* 改变文字颜色 */
    border-left: 12px solid yellow;
}
```



图 6.7　水平布局列表结构





Note



视频讲解

> **提示：** 为了解决浮动存在的列表框自动收缩问题，上面示例把列表框（<ul>标签）、列表项（<li>标签）和超链接（<a>标签）都定义为浮动显示（float: left;），简化列表框自动收缩所带来的复杂问题。

6.2.3 设计菜单样式

本节案例通过 CSS 背景图设计一款菜单样式，定义当鼠标指针移到菜单上时会显示另一种背景图，演示效果如图 6.8 所示。



图 6.8　设计个人网站菜单演示效果

【操作步骤】

第 1 步，启动 Dreamweaver，新建网页并保存为 index.html。打开网页文档，设计如下导航菜单结构。

```
<ul class="menu">
    <li class="top"><a href="#" class="top_link"><span>首页</span></a></li>
    <li class="top"><a href="#" class="top_link"><span>我的相册</span></a></li>
    <li class="top"><a href="#" class="top_link"><span>我的日志</span></a></li>
    <li class="top"><a href="#/" class="top_link"><span>我的音乐盒</span></a></li>
    <li class="top"><a href="#" class="top_link"><span>我的介绍</span></a></li>
    <li class="top"><a href="#" class="top_link"><span>留言本</span></a></li>
</ul>
```

在这个导航菜单中，包含两层结构，外层的<ul>标签控制导航总体样式，内层的<li>标签控制每个菜单项的样式。

第 2 步，为了与页面其他模块的列表结构进行区分，在该导航菜单中定义外层的<ul>标签类名为 menu，内层的<li>标签类名为 top，每个选项中包含的超链接类名为 top_link。

第 3 步，新建 CSS 样式表文件，命名为 style.css，保存到 images 文件夹中，然后在页面头部区域导入该样式表，代码如下。

```
<link rel="stylesheet" href="images/style.css" type="text/css" />
```

第 4 步，设计菜单框架样式，控制整个导航菜单外观。这里主要通过类选择器.menu 实现，样式细节包括：设置内外边距，固定高度为 40 像素，清除列表框默认样式（如项目符号、缩进），定义背景效果，设置字体样式，设计外框为相对定位（position:relative;），代码如下。



```
.menu {padding:0 0 0 32px; margin:0; list-style:none; height:40px; background:#fff url(button1a.gif) repeat-x; position:relative; font-family:arial, verdana, sans-serif; margin-top:50px;}
```





其中，position:relative;声明对于整个案例效果的影响最为关键，它能够约束内部结构的布局，关于 position 属性的深入讲解请参阅后面章节。

第 5 步，以包含选择器的方式匹配每个列表项，然后定义每个列表项以块状显示，并向左浮动，实现横向并列显示效果，代码如下。

```
.menu li.top {display:block; float:left; position:relative;}
```

注意，这里使用包含选择器，限制匹配范围为导航菜单框内，然后使用指定范围类样式，以便提高类样式的优先级，以及确定样式应用的标签类型范围。

第 6 步，通过多层包含选择器定义选项中超链接的样式。设置每个超链接以块状、向右浮动显示，然后定义高度，与导航栏同高，定义菜单内字体样式等，代码如下。

```
.menu li a.top_link {display:block; float:left; height:40px; line-height:33px; color:#bbb; text-decoration:none; font-size:11px; font-weight:bold; padding:0 0 0 12px; cursor:pointer;}
```

第 7 步，继续以多层包含选择器定义超链接中包含的 span 元素样式，该样式与超链接样式相似，代码如下。

```
.menu li a.top_link span {float:left; font-weight:bold; display:block; padding:0 24px 0 12px; height:40px;}
```

第 8 步，以伪类选择器方式定义鼠标经过超链接的样式。该效果主要包含两个样式，它们分别定义 a 和 span 元素样式。在样式中主要重设背景图像和位置，代码如下。

```
.menu li a.top_link:hover {color:#000; background: url(button4.gif) no-repeat;}
.menu li a.top_link:hover span {background:url(button4.gif) no-repeat right top;}
```

第 9 步，以选择器分组的方式定义导航菜单中的公共样式，如初始化 ul 默认效果，考虑到在不同位置、不同状态下的 ul 样式，这里采用了选择器分组的方式统一进行控制，代码如下。

```
.menu ul,
.menu :hover ul ul,
.menu :hover ul :hover ul ul,
.menu :hover ul :hover ul :hover ul ul,
.menu :hover ul :hover ul :hover ul :hover ul ul {position:absolute; left:-9999px; top:-9999px; width:0; height:0; margin:0; padding:0; list-style:none;}
```

其他样式以及整个案例效果请参阅本节实例源代码。

6.2.4　设计导航条



本节案例利用背景图像设计导航条，通过背景图像的衬托使导航列表显得醒目、有立体感，具





Note

有 Vista 系统的超酷效果，如图 6.9 所示。

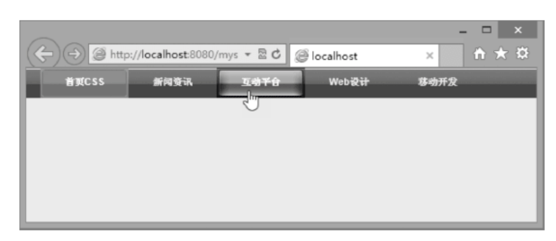


图 6.9　使用背景图像设计导航条

【操作步骤】

第 1 步，启动 Photoshop，设计背景图像。可以在 Photoshop 中使用渐变工具绘制椭圆条状形，然后利用选取工具分别选取上下部分，分别应用曲线工具把上半部分调亮，把下半部分调暗即可，也可以利用图层样式，并借助叠加等功能完善这种立体效果。图像设计得越形象，所设计的导航条也就越逼真。本案例设计了 3 个背景图像，如图 6.10 所示。



正常显示背景图像　　鼠标指针经过时显示的背景图像　　当前按钮显示背景图像

图 6.10　设计导航条所用的背景图像

第 2 步，新建文档，构建该导航条的 HTML 结构框架，为了方便控制背景图像，在列表项<li>标签中包含了一个辅助元素 b，代码如下。

```
<ul class="menu">
    <li class="current"><a href="#"><b>首页 CSS</b></a></li>
    <li><a href="#"><b>新闻资讯</b></a></li>
    <li><a href="#"><b>互动平台</b></a></li>
    <li><a href="#"><b>Web 设计</b></a></li>
    <li><a href="#"><b>移动开发</b></a></li>
</ul>
```

第 3 步，在头部区域<head>标签内添加<style>标签，新建内部样式表。使用 CSS 控制导航条显示，代码如下。

```
body { background-color: #dee0ff; padding:0; margin:0; }
.menu {
    padding:0 0 0 1em;                  /* 在导航条左侧增加 1 个字大小的间距 */
    margin:0;                           /* 清除默认缩进样式 */
    list-style:none;                    /* 清除项目符号 */
    height:35px;                        /* 固定导航条高度 */
    background:url(images/bg3.gif);     /* 定义导航条的背景图像 */
}
```





第 4 步，定义列表项浮动显示，设计水平显示，代码如下。

```
.menu li {float:left;}
```

第 5 步，定义菜单项的显示样式，设置超链接以块状显示，并固定大小，设置菜单文本显示属性，代码如下。

```
.menu li a {
    display:block;                              /* 块状显示 */
    float:left;                                 /* 向左浮动 */
    height:35px;                                /* 与导航条同高 */
    line-height:33px;                           /* 垂直对齐文本 */
    color:#FFFF00;                              /* 设置粉红色字体颜色 */
    text-decoration:none;                       /* 清除默认样式下划线 */
    font-family:arial, verdana, sans-serif;     /* 字体属性 */
    text-align:center;                          /* 水平对齐文本 */
    padding:0 0 0 14px;                         /* 增加左侧空隙 */
    cursor:pointer;                             /* 定义手形鼠标指针 */
    font-size:11px;                             /* 字体大小 */
}
```

第 6 步，设置 a 元素所包含的辅助元素也为块状浮动显示，这样为后面进行控制提供保障，代码如下。

```
.menu li a b {
    float:left;                     /* 向左浮动 */
    display:block;                  /* 块状显示 */
    padding:0 28px 0 14px;          /* 增加左右两侧的内边距 */
}
```

第 7 步，定义当前菜单的显示样式，代码如下。所谓当前菜单就是被激活的菜单，也就是当前页面为当前菜单指向的链接。定义当前菜单样式是为了更好地区分菜单状态。

```
.menu li.current a {
    color:#fff;                                 /* 白色字体 */
    background:url(images/left3.gif);           /* 定义当前菜单的背景图像 */
}
.menu li.current a b { /* 定义当前菜单的背景图像 */
    background:url(images/left3.gif) no-repeat right top;
}
```

上面分别在 a 和 b 元素中同时定义相同的背景图像，利用背景图像重叠来伪造圆角按钮效果。





Note

如果仅定义 a 元素的背景图像，这时会看见当前按钮右侧显示为直角效果，而不是与左侧对应的圆角，如图 6.11 所示。



图 6.11　仅定义一个背景图像的效果

通过为两个重合的元素定义相同的背景图像，一个从左侧开始向右侧延伸，另一个从右侧向左侧平铺。由于是背景图像，中间多余的区域会被自动隐藏，这样就给人一种错觉，所显示的按钮是圆角效果，演示示意图如图 6.12 所示。



图 6.12　圆角背景图像的设计示意图

第 8 步，定义鼠标指针经过时的样式。设计思路与上面的方法相同，就不再重复说明，代码如下。

```
.menu li.current a:hover {
    color:#fff;                                /* 白色字体 */
    background: url(images/left 3.gif);        /* 定义鼠标指针经过的背景图像 */
    cursor:default;                            /* 定义手形鼠标样式 */
}
.menu li.current a:hover b {                   /* 定义鼠标指针经过的背景图像 */
    background:url(images/left 3.gif) no-repeat right top;
}
```



视 频 讲 解

6.2.5　设计下拉菜单

本节案例演示如何利用 CSS 技术，并适当借助 JavaScript 脚本控制下拉菜单的显示和隐藏，演示效果如图 6.13 所示。



图 6.13　下拉菜单演示效果



【操作步骤】

第 1 步，首先构建一个下拉菜单结构，代码如下。

```
<ul  id="nav">
    <li class="menu2" ><a href="#">查看样式表 CSS</a>
        <ul class="list">
            <li><a href="#">子菜单 1</a></li>
            <li><a href="#">子菜单 2</a></li>
            <li><a href="#">子菜单 3</a></li>
        </ul>
    </li>
    <li class="menu2" ><a href="#">CSS 参考资料</a>
        <ul class="list">
            <li><a href="#">子菜单 1</a></li>
            <li><a href="#">子菜单 2</a></li>
        </ul>
    </li>
    <li class="menu2" ><a href="#">常见问题</a>
        <ul class="list">
            <li><a href="#">子菜单 1</a></li>
        </ul>
    </li>
    <li class="menu2" ><a href="#">投稿</a> </li>
    <li class="menu2" ><a href="#">翻译文件</a> </li>
</ul>
```

这是一个嵌套的二级项目列表，外层项目列表负责组织主菜单，内层项目列表负责管理子菜单，当然也可以在此基础上进一步扩展。该项目列表结构插入案例模板的头部区域底部。

第 2 步，新建内部样式表，然后开始编写样式。为该导航条结构定义样式，代码如下。

```
#nav {
    position:absolute;              /* 绝对定位 */
    z-index:1; /* 定义层叠顺序，避免页面中部的层（左侧导航条）覆盖本菜单 */
    left: 0px;                      /* 左侧距离 */
    top: 124px;                     /* 顶部距离 */
    width:700px;                    /* 宽度 */
    height:30px;                    /* 高度 */
    padding:0px 4px;                /* 增加左右内边距 */
}
```




Note

```
html>/**/body #nav { /*  兼容非 IE 浏览器  */
    left: 40px;                    /*  左侧距离  */
    top: 112px;                    /*  顶部距离  */
}
```

这里使用了一个兼容技术，选择符 html>/**/body #nav 能够限制该样式仅在非 IE 浏览器中被解析。由于 IE 浏览器与非 IE 浏览器在解析复杂定位时（包含多个定位包含框的时候），会存在判断标准的不同，导致解析结果存在很大差异。

第 3 步，清除项目列表和超链接的默认样式，代码如下。

```
#nav ul { margin:0px; padding:0px; }          /*  清除缩进  */
#nav li a { text-decoration:none; }           /*  清除下划线  */
```

第 4 步，让列表项浮动并列显示，并定义主菜单的显示样式，代码如下。

```
#nav li {
    list-style:none;                      /*  清除项目符号  */
    text-align:center;                    /*  居中对齐  */
    font-weight:bold;                     /*  加粗显示  */
    float:left;                           /*  向左浮动*/
}
```

第 5 步，定义下拉子菜单的显示样式，代码如下。

```
#nav .list {
    line-height:20px;                     /*  行高  */
    text-align:left;                      /*  左对齐  */
    padding:2px;                          /*  内边距  */
    font-weight:normal;                   /*  正常字体，不加粗显示  */
    }
#nav .list a {
    color:#FF3AC1;                        /*  下拉子菜单中超链接字体颜色  */
    text-decoration:none;                 /*  清除下划线  */
    float:left;                           /*  浮动超链接显示，这样可以定义宽和高  */
    width:100px;                          /*  超链接的宽度  */
    padding:3px 5px 0px 5px;              /*  内边距  */
}
```

第 6 步，定义鼠标指针经过子菜单时显示的样式，代码如下。

```
#nav .list a:hover {
    color:white;                          /*  字体颜色  */
```





```
    padding:3px 3px 0px 20px;          /* 内边距 */
    width:88px;                        /* 宽度 */
    background-color:#FF3AC1;          /* 背景色 */
}
```

第 7 步，定义两个类样式，设计当鼠标指针经过和离开主菜单时所要应用的样式，代码如下。

```
#nav .menu1 {
    width:120px;                       /* 主菜单的宽度 */
    height:auto;                       /* 主菜单的高度，自动 */
    margin:6px 4px 0px 0px;            /* 增加外边距 */
    border:1px solid #FF3AC1;          /* 设置一个边框 */
    background-color:#F1FBEC;          /* 定义背景色 */
    color:#FF3AC1;                     /* 字体颜色 */
    padding:6px 0px 0px 0px;           /* 定义内边距 */
    cursor:hand;                       /* 定义鼠标指针样式，手形 */
    overflow-y:hidden;                 /* 隐藏 y 轴超出的区域 */
    filter:Alpha(opacity=70);          /* 在 IE 下设置透明度 */
    -moz-opacity:0.7;                  /* 在非 IE 下设置透明度 */
}
#nav .menu2 {
    width:120px;
    height:18px;
    margin:6px 4px 0px 0px;
    background-color:#F5F5F5;
    color:#999999;
    border:1px solid #EEE8DD;
    padding:6px 0px 0px 0px;
    overflow-y:hidden;
    cursor:hand;
}
```

第 8 步，定义完毕两个类样式之后，就可以在主菜单标签中的属性事件中引用这两个类样式，代码如下。

```
<li class="menu2" onMouseOver="this.className='menu1'" onMouseOut="this.className='menu2'">
```

6.2.6　设计折叠导航面板

本例演示如何利用 CSS 设计一款折叠导航面板，演示效果如图 6.14 所示。实际上，本例就是一





个下拉菜单，只不过仅包含一个菜单项目，如果读者感兴趣，可以扩展多个列表项目，那样就会形成一个下拉菜单样式。



收缩状态



展开状态

图 6.14　折叠导航面板

【操作步骤】

第 1 步，构建导航结构。这是一个嵌套的列表结构，外层为无序列表，内层是定义列表。为了能够兼容早期 IE 浏览器，这里使用 IE 条件语句做向下兼容修补，使 IE 6 和 IE 7 能够正确响应鼠标行为，代码如下。

```
<ul id="menu">
    <li>
        <!--[if lte IE 6]><a href="#nogo"><table><tr><td><![endif]-->
        <dl>
            <dt class="orange"><a href="#">网站导航</a></dt>
            <dd><a href="#">首页</a></dd>
            <dd><a href="#">新闻资讯</a></dd>
            <dd><a href="#">关于我们</a></dd>
            <dd class="last"><a href="#">互动平台</a></dd>
        </dl>
        <!--[if lte IE 6]></td></tr></table></a><![endif]-->
    </li>
</ul>
```

第 2 步，新建内部样式表，然后开始编写样式。为外层列表结构定义样式，代码如下。

```
#menu {/* 整体设置*/
    margin: 0; padding: 0;
    list-style-type: none;
    font: 14px Arial;
}
#menu li {
```





```
    float: left;
    padding: 0; margin: 0 1px 0 0;
    width: 150px;
}
```

第 3 步，为内层的列表结构定义样式，代码如下。

```
#menu li dl {/*  设置菜单项*/
    width: 150px;/*ie6*/
    margin: 0; padding: 0 0 10px 0;
    background: #cb6 url(images/bottom.gif) no-repeat bottom left;
}
#menu li dt a, #menu li dd a { display: block; }
```

第 4 步，设置菜单项 dt 的样式，代码如下。

```
#menu li dt {
    margin: 0; padding: 5px;
    text-align: center;
    border-bottom: 1px solid #b00;
}
#menu li dt.orange { background: #fa5 url(images/top.gif) no-repeat top left; }
#menu li dt.yellow { background: #ee5 url(images/top.gif) no-repeat top left; }
#menu li dt.green { background: #5e5 url(images/top.gif) no-repeat top left; }
#menu li dt.blue { background: #5cf url(images/top.gif) no-repeat top left; }
#menu li dt a, #menu li dt a:visited {
    display: block;
    color: #333; text-decoration: none;
}
```

第 5 步，设置菜单项 dd 的样式，代码如下。

```
#menu li dd {
    margin: 0; padding: 0;
    color: #fff; background: #47a;
}
#menu li dd.last { border-bottom: 1px solid #b00; }
#menu li dd a, #menu li dd a:visited {
    display: block;
    color: #fff; text-decoration: none;
    padding: 4px 5px 4px 20px;
```




Note

```
        background: #47a url(images/arrow.gif) no-repeat 10px 10px;
        height: 1em;
}
```

第 6 步，默认状态下关闭子菜单，代码如下。

```
#menu li dd { display: none; }
```

第 7 步，设置鼠标响应，代码如下。

```
#menu li:hover dd, #menu li a:hover dd { display: block; }
#menu li:hover, #menu li a:hover { border: 0; }/*ie6*/
#menu li dd a:hover { background: #147; color: #9cf;}
```

6.2.7 设计带提示信息的菜单

本例设计一款自动显示提示信息的菜单，这是一个很实用的效果。当鼠标指针经过某一个菜单项的时候，在菜单右侧会出现一个矩形区域，里面写着对正在经过的菜单项的说明文字，效果如图 6.15 所示。



图 6.15 带提示信息的菜单

【操作步骤】

第 1 步，构建菜单结构，代码如下。

```
<div id="menu">
    <a href="#">
            <span class="left"></span>首页<span class="right"></span>
            <span class="intro">欢迎光临本站</span>
     </a>
    <a href="#">
            <span class="left"></span> 网页设计<span class="right"></span>
            <span class="intro">这里有很多设计师</span>
    </a>
    <a href="#">
            <span class="left"></span>联系我们<span class="right"></span>
```



```
            <span class="intro">QQ：66666666</span>
        </a>
    </div>
```



对于每一个菜单项的 a 元素，分别再增加一个 span 元素，里面包裹相应的说明文字。在默认状态下，把这些说明信息隐藏起来，当鼠标指针经过某一个菜单项的时候，再打开该 span 即可。

第 2 步，新建内部样式表，然后开始编写样式，定义菜单外框样式，代码如下。

```
#menu {                                  /*对 menu 层设置*/
    font-family: Arial;                  /*字体*/
    font-size: 16px;                     /*字号*/
    width: 140px;                        /*宽度*/
    margin: 0;                           /*清除边界*/
    border: solid 1px #ccc;              /*灰色细边框*/
}
```

第 3 步，设计超链接的样式，代码如下。

```
#menu a, #menu a:visited {
    text-decoration: none;               /*文字无下划线*/
    text-align: center;                  /*文字水平居中对齐*/
    color: #c00;                         /*红色文字*/
    display: block;                      /*设置为块级元素*/
    padding: 4px;                        /*内边距*/
    background-color: #fff;              /*背景色*/
    border: solid 1px #fff;              /*与背景色相同边框，防止跳动*/
    position: relative;                  /*使用相对定位*/
    width: 130px;
}
```

第 4 步，在默认状态下，先隐藏提示信息文本，代码如下。

```
#menu a span { display: none; }
```

第 5 步，设计当鼠标指针经过菜单项目时，显示左右箭头特效，代码如下。

```
#menu a:hover { border-color: #c00;}          /*边框颜色为红色*/
#menu a:hover span {
    display: block;                           /*设置为块级元素*/
    position: absolute;                       /*使用绝对定位*/
    height: 0;                                /*高度为 0*/
    width: 0;                                 /*宽度为 0*/
```




Note

```
        overflow: hidden;                           /*防止溢出*/
        border: solid 8px #fff;                     /*设置默认的边框样式*/
        top: 4px;                                   /*竖直方向的定位*/
}
#menu a:hover span.left { border-left-color: #c00; left: 8px;}
#menu a:hover span.right { border-right-color: #c00; right: 8px;}
```

第 6 步，设计当鼠标指针经过时，显示提示信息文本，并定位到菜单项的右侧显示，代码如下。

```
#menu a:hover span.intro {
        font-size: 12px;
        display: block; padding: 5px;
        position: absolute;                             /*绝对定位*/
        left: 150px; top: 0px;
        width: 100px; height: auto;
        background-color: #eee; color: #000;
        border: 1px dashed #234;
}
```


视频讲解

6.2.8 设计排行榜

音乐排行榜，主要体现的是当前某个时间段中某些歌曲的排名情况。如图 6.16 所示为本节案例的效果图，该例展示了音乐排行榜在网页中的基本设计样式。



图 6.16　音乐排行榜栏目

【操作步骤】

第 1 步，新建网页，保存为 index.html，在<body>标签内编写如下结构代码，构建 HTML 文档。

```
<div class="music_sort">
<h1>音乐排行榜</h1>
<div class="content">
```



```
            <ol>
                <li><strong>浪人情歌</strong> <span>伍佰</span></li>
                <li><strong>K 歌之王</strong> <span>陈奕迅</span></li>
                <li><strong>心如刀割</strong> <span>张学友</span></li>
                <li><strong>零（战神 主题曲）</strong> <span>柯有伦</span></li>
                <li><strong>双子星</strong> <span>光良</span></li>
                <li><strong>离歌</strong> <span>信乐团</span></li>
                <li><strong>海阔天空</strong> <span>信乐团</span></li>
                <li><strong>天高地厚</strong> <span>信乐团</span></li>
                <li><strong>边走边爱</strong> <span>谢霆锋</span></li>
                <li><strong>想到和做到的</strong> <span>马天宇</span></li>
            </ol>
        </div>
    </div>
```

第 2 步，厘清设计思路。首先，将默认的显示效果与通过 CSS 样式修饰过的显示效果进行对比，如图 6.17 所示，可以发现两者的不同之处。

☑　文字的大小。

☑　榜单排名序号的样式。

☑　背景色和边框色的修饰。



图 6.17　CSS 样式修饰后（左）与无 CSS 样式修饰（右）的对比

通过对比可见，数字序号已经不再是普通的常见文字了，而是经过特殊处理的文字效果，换言之，必须使用图片才可以达到预期效果。这个数字图片在列表中的处理方式也就是本例需要讲解的部分，在讲解之前先思考以下两个问题。

☑　10 个数字，也就是 10 张图片，可不可以将这 10 张图片合并成 1 张图片？

☑　将 10 张图片合并成 1 张图片，但 HTML 结构中又没有针对每个列表<li>标签添加 Class 类名，怎么将图片指定到相对应的排名中？

第 3 步，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，准备编写样式。





第 4 步，针对第 2 步分析的两个主要问题，编写如下 CSS 样式。

```
.music_sort {
    width:200px;
    border:1px solid #E8E8E8;
}
.music_sort * {/* 清除.music_sort 容器中所有元素的默认内补丁和外补丁，并设置文字相关属性 */
    margin:0;
    padding:0;
    font:normal 12px/22px "宋体", Verdana,Lucida, Arial, Helvetica, sans-serif;
}
.music_sort h1 {
    height:24px;
    text-indent:10px; /* 标题文字缩进，增加空间感 */
    font-weight:bold;
    color:#FFFFFF;
    background-color:#999999;
}
.music_sort ol {
    height:220px;           /* 固定榜单列表的整体高度 */
    padding-left:26px;      /* 利用内补丁增加 ol 容器的空间显示背景图片 */
    list-style:none;        /* 去除默认的列表修饰符 */
    background:url(images/number.gif) no-repeat 0 0;
}
.music_sort li {
    width:100%;
    height:22px;
    list-style:none;        /* 去除默认的列表修饰符 */
}
.music_sort li span {color:#CCCCCC; /* 将列表中的歌手名字设置为灰色 */}
```

这段 CSS 样式就是为了实现最终效果而写的，代码设计思路如下。

将有序列表<ol>标签的高度属性值设定一个固定值，这个固定值为列表<li>标签的 10 倍；并将列表所有的默认样式修饰符取消；利用有序列表<ol>标签中增加左补丁的空间显示合并后的数字背景图。

简单的方法代替了给不同的列表<li>标签添加不同背景图片的步骤。但这种处理方式的缺陷就是必须调整好背景图片中 10 个数字图片之间的间距，而且如果增加了每个列表<li>标签的高度，那么就需要重新修改背景图片中 10 个数字图片之间的间距。







6.2.9　设计图文列表栏目

图文列表的结构就是将列表内容以图片的形式在页面中显示，简单理解就是图片列表信息附带简短的文字说明。在图中展示的内容主要包含列表标题、图片和图片的相关说明文字。

【操作步骤】

第 1 步，新建网页，保存为 index.html，在<body>标签内编写如下结构，构建 HTML 文档。

```
<div class="pic_list">
    <h3>爱秀</h3>
    <div class="content">
        <ul>
            <li><a href="#"><img src="images/1.jpg" alt="美女个性搞怪自拍">美女个性搞怪自拍</a></li>
            <li><a href="#"><img src="images/2.jpg" alt="绝对阳光的清纯小妹">绝对阳光的清纯小妹</a></li>
            <li><a href="#"><img src="images/3.jpg" alt="漂亮美女的可爱外拍">漂亮美女的可爱外拍</a></li>
            <li><a href="#"><img src="images/4.jpg" alt="可爱美女的艺术照">可爱美女的艺术照</a></li>
            <li><a href="#"><img src="images/5.jpg" alt="漂亮美女娇美自拍">漂亮美女娇美自拍</a></li>
            <li><a href="#"><img src="images/6.jpg" alt="清纯迷人的黄毛丫头">清纯迷人的黄毛丫头</a></li>
        </ul>
    </div>
</div>
```

第 2 步，梳理结构。对于列表的内容不再细讲，这个列表的 HTML 结构如图 6.18 所示，结构层次清晰而富有条理。

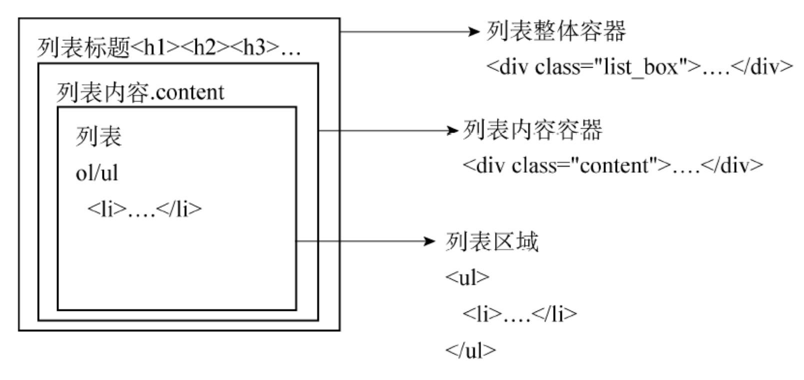


图 6.18　列表结构的分析示意图

该结构不仅在HTML代码中能很好地体现页面结构层次，且方便后期使用CSS样式对其进行设计。

第 3 步，梳理设计思路。图文列表的排列方式最讲究的是宽度属性的计算。横向排列的列表，当整体的列表（有序列表或者无序列表）横向空间不足以将所有列表横向显示时，浏览器会将列表换行显示，如图 6.19 所示。所以只有将宽度计算正确，才有足够空间将所有列表横向排列显示并且不会产生空间的浪费。




Note



图 6.19　列表宽度计算不正确导致的结果

第 4 步，设计栏目宽度。在本例中，每张图片的宽度为 134px，左右内补丁分别为 3px，左右边框分别为 1px 宽度的线条，且图片列表与图片列表之间的间距为 15px（即右外补丁为 15px），根据盒模型的计算方式，最终列表<li>标签的盒模型宽度值为 1px+3px+134px+3px+1px+15px=157px，因此图文列表区域总宽度值为 157px×6=942px。

第 5 步，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，准备编写样式。

第 6 步，编写图文列表区域的相关 CSS 样式代码。

```
.pic_list .content {
    width:942px;
    height:150px;
    overflow:hidden;          /* 设置图文列表内容区域的宽度和高度，超过部分隐藏 */
    padding:22px 0 0 15px;    /* 利用内补丁增加图文列表内容区域与其他元素的间距 */
}
.pic_list .content li {
    float:left;
    width:142px;
    margin-right:15px;        /* 列表<li>标签设置浮动后，所有列表将根据盒模型的计算方式计算列
                              表宽度，并且并排显示 */
    display:inline;           /* 设置浮动并且增加了左右外补丁后，IE 6 会产生双倍间距的 bug，利
                              用该属性解决 */
    }
```

.pic_list .content 作为图文列表内容区域，增加相应的内补丁使其与整体之间有空间感，这个是视觉效果中必然会处理的一个问题。

由于.pic_list .content li 具有浮动属性，并且有左右外补丁中一个外补丁属性，在 IE 6 浏览器中会产生双倍间距的 bug 问题。而神奇的是，添加 display:inline 可以解决该问题，并且不会对其他浏览器产生任何影响。

第 7 步，主要的内容设置成功之后，就可以对图文列表的整体效果做 CSS 样式的修饰，例如图文列表的背景和边框以及图文列表标题的高度、文字样式和背景等。



```
.pic_list {
    width:960px;                                /*  设置图文列表整体的宽度 */
    border:1px solid #D9E5F5;                   /*  添加图文列表的边框 */
    background:url(images/wrap.jpg) repeat-x 0 0;   /*  添加图文列表整体的背景图片 */
}
.pic_list * {/*  重置图文列表内部所有基本样式 */
    margin:0;
    padding:0;
    list-style:none;
    font:normal 12px/1.5em "宋体", Verdana,Lucida, Arial, Helvetica, sans-serif;
}
.pic_list h3 { /*  设置图文列表标题的高度、行高、文字样式和背景图片 */
    height:34px; line-height:34px; font-size:14px;
    text-indent:12px;
    font-weight:bold; color:#223A6D;
    background:url(images/h3bg.jpg) no-repeat 0 0;
}
```

第 8 步，调整图文列表信息细节以及对用户体验的把握，例如图片的边框、背景和文字的颜色等，并且还要考虑当用户控制鼠标经过图片时，为了能更好地体现视觉效果，给用户一个全新的体验，添加当鼠标指标经过图片列表信息时图片以及文字的样式变化，代码如下。

```
.pic_list .content li a {
    display:block;              /*  将内联元素 a 转换为块元素使其具备宽高属性 */
    width:142px;                /*  设置转换为块元素后的 a 元素的宽度 */
   text-align:center;           /*  文本居中显示 */
    text-decoration:none;       /*  文本下划线 */
    color:#333333;              /*  文本的颜色 */
}
.pic_list .content li a img {
    display:block;              /*  当设置为块元素时，可以解决 IE 6 中图片底部几个空白像素的 bug */
    width:134px;
    height:101px;
    padding:3px;                /*  设置图片的宽高属性以及内补丁属性 */
    margin-bottom:8px;          /*  将图片的底部外补丁设置为 8px，使其与文字之间产生一定间距 */
    border:1px solid #CCCCCC;
    background-color:#FFFFFF; /*  背景颜色将通过内补丁的空间显示 */
}
.pic_list .content li a:hover {
```




```
        text-decoration:underline;
        color:#CC0000;              /* 当鼠标指针经过图文列表时，文字有下划线并且改变颜色 */
    }
    .pic_list .content li a:hover img {
        background-color:#22407E;   /* 当鼠标指针经过图文列表时，图片的背景颜色改变 */
    }
```

第 9 步，保存页面，然后在浏览器中预览，演示效果如图 6.20 所示。



图 6.20　图文信息列表页面效果



视 频 讲 解

6.2.10　设计选项卡

本例使用定义列表结构设计一个选项卡面板，主要使用了以下两个技巧。

☑　利用绝对定位来定义 dt 元素内容的显示位置，即定义显示在 dd 元素之上。

☑　利用锚技术为每个 ul 列表定义锚记，然后利用 dt 中的超链接实现内容切换。

【操作步骤】

第 1 步，启动 Dreamweaver，新建 HTML5 文档，保存为 index.html。

第 2 步，在页面中构建 HTML 结构。切换到代码视图，在<body>标签内手动输入下面代码。

```
    <dl><!-- 定义列表 -->
        <dt><a href="#a" title="">焦点新闻</a><a href="#b" title="">国内新闻</a><a href="#c" title="">国际新闻</a></dt><!-- 定义列表标题 -->
        <dd><!-- 定义列表说明 -->
            <ul id="a"><!-- 内嵌无序列表 -->
                <li>•<a href="" title="">焦点新闻 1</a></li>
                <li>•<a href="" title="">焦点新闻 2</a></li>
                <li>•<a href="" title="">焦点新闻 3</a></li>
                <li>•<a href="" title="">焦点新闻 4</a></li>
                <li>•<a href="" title="">焦点新闻 5</a></li>
            </ul>
```



```
        <ul id="b"><!-- 内嵌无序列表 -->
            <li>•<a href="" title="">国内新闻 1</a></li>
            <li>•<a href="" title="">国内新闻 2</a></li>
            <li>•<a href="" title="">国内新闻 3</a></li>
            <li>•<a href="" title="">国内新闻 4</a></li>
            <li>•<a href="" title="">国内新闻 5</a></li>
        </ul>
        <ul id="c"><!— 内嵌无序列表 -->
            <li>•<a href="" title="">国际新闻 1</a></li>
            <li>•<a href="" title="">国际新闻 2</a></li>
            <li>•<a href="" title="">国际新闻 3</a></li>
            <li>•<a href="" title="">国际新闻 4</a></li>
            <li>•<a href="" title="">国际新闻 5</a></li>
        </ul>
    </dd>
</dl>
```

第 3 步，在<head>标签内输入<style type="text/css">，定义一个内部样式表，然后在<style>标签内手动输入下面的样式代码。

```
dl {/*定义列表属性*/
    position:relative;                          /*相对定位，定义一个包含块*/
    width:240px;                                /*自定义宽*/
    height:200px;                               /*自定义高*/
    }
dt {/*定义列表标题属性*/
    position:absolute;                          /*绝对定位，根据上级元素 dl 包含块进行精确定位*/
    left:-2px;                                  /*定位在 dl 元素包含块左侧外边 2 像素位置，使用负值可以定义
                                                在外边*/
    top:-1.5em;                                 /*定位在 dl 元素包含块顶部外边 1.5em 位置*/
    }
dt a {/*定义列表标题内链接属性*/
    display:block;                              /*块状显示*/
    float:left;                                 /*浮动左对齐*/
    margin:1px;                                 /*定义边距*/
    width:78px;                                 /*定义标题栏宽度*/
    text-align:center;                          /*文本居中*/
    font:bold 12px/1.8em "宋体",sans-serif;     /*定义字体属性*/
    color:#fff;                                 /*定义字体颜色*/
```




Note

```
        text-decoration:none;                /*清除下划线*/
        background:#666;                     /*定义背景色*/
    }
dt a:hover {/*定义列表标题内链接鼠标指针经过属性 */
        background:orange;                   /*改变背景色*/
            }
dd {/*定义列表说明属性*/
        margin:0;                            /*清除预定义边界*/
        width:240px;                         /*自定义宽，与父元素宽度保持一致*/
        height:200px;                        /*自定义高，与父元素高度保持一致*/
        overflow:hidden;                     /*隐藏超出区域*/
        border:1px solid #999;               /*定义边框*/
    }
ul {/*定义无序列表属性*/
        margin:0;                            /*清除边界预定义值*/
        padding:6px 0;                       /*自定义补白*/
        width:240px;                         /*自定义宽，与上面宽度保持一致*/
        height:200px;                        /*自定义高，与上面高度保持一致*/
        list-style:none;                     /*清除样式预定义值*/
    }
li {/*定义无序列表项属性*/
        width:230px;                         /*自定义宽，要小于父元素定义的宽度*/
        font:12px/1.8em "宋体",sans-serif;   /*字体属性*/
        white-space:nowrap;                  /*禁止换行显示*/
        overflow:hidden;                     /*隐藏超出区域内容*/
    }
```

第 4 步，保存文档，按 F12 功能键，在浏览器中预览，效果如图 6.21 所示。

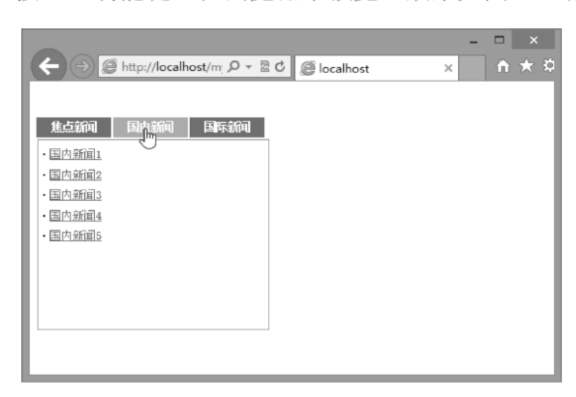


图 6.21　选项卡面板效果







6.2.11　设计多级菜单

多级菜单可以包含更多的链接，在有限的空间内实现更多导航选项。在没有使用 CSS 之前，用户习惯使用 JavaScript 来实现多级菜单。多级菜单的样式也比较丰富，如平行式、垂直式、并列式、层叠式等。本例用纯 CSS 设计的多级菜单方便易用，支持所有浏览器。用户可以将本例作为模板推演出更多样式的纯 CSS 多级菜单。

【操作步骤】

第 1 步，启动 Dreamweaver，新建 HTML5 文档，保存为 index1.html。

第 2 步，在页面中构建 HTML 导航框架结构。切换到代码视图，在<body>标签内手动输入如下代码。

```
<div class="menu">
    <ul>
    <li><a href="#">菜单一      &#187;
        <!--[if IE 7]><!-->
        </a>
        <!--<![endif]-->
        <!--[if lte IE 6]><table><tr><td><![endif]-->
        <ul>
        <li><a href="#">菜单一      &#187;
            <!--[if IE 7]><!-->
            </a>
            <!--<![endif]-->
            <!--[if lte IE 6]><table><tr><td><![endif]-->
            <ul>
            <li><a href="#">菜单一 1    &#187;
                <!--[if IE 7]><!-->
                </a>
                <!--<![endif]-->
                <!--[if lte IE 6]><table><tr><td><![endif]-->
                <ul>
                <li><a href="#">菜单一 1-1</a></li>
                <li><a href="#">菜单一 1-2</a></li>
                <li><a href="#">菜单一 1-3</a></li>
                </ul>
                <!--[if lte IE 6]></td></tr></table></a><![endif]-->
            </li>
            <li><a href="#">菜单一 2    &#187;
                <!--[if IE 7]><!-->
```





Note

```
                    </a>
                    <!--<![endif]-->
                    <!--[if lte IE 6]><table><tr><td><![endif]-->
                    <ul>
                    <li><a href="#">菜单一 2-1</a></li>
                    <li><a href="#">菜单一 2-2</a></li>
                    <li><a href="#">菜单一 2-3</a></li>
                    </ul>
                    <!--[if lte IE 6]></td></tr></table></a><![endif]-->
                </li>
                </ul>
                <!--[if lte IE 6]></td></tr></table></a><![endif]-->
            </li>
            …
        </ul>
    </div>
```

第 3 步，在<head>标签内输入<style type="text/css">，定义一个内部样式表，然后在<style>标签内手动输入下面的样式代码。

```
.menu {/*定义多级菜单具有更高层叠顺序*/
    z-index: 1000;
    font-size: 12px;
    }
.menu ul {/*定义一级列表属性*/
    padding: 0;                 /*清除补白默认值*/
    margin: 0;                  /*清除边界默认值*/
    list-style-type: none;      /*清除默认样式*/
    width: 100px;               /*定义宽度*/
    position: relative;         /*相对定位，定义包含块*/
    border: 1px solid #fff;     /*定义白色边框*/
    }
.menu li {/*定义列表项属性*/
    background: #AFDD22;        /*定义背景色*/
    height: 26px;               /*定义显示高度*/
    position: relative;         /*相对定位，定义包含块，实现精确定位*/
    }
* html .menu li {/*兼容 Hacks，定义在 IE 6 及更低版本中列表项属性*/
    float: left;                /*向左浮动*/
```





```
    margin-left: -16px;          /*左边界取负*/
    margin-lef\t: 0;             /*兼容 Hacks，定义在 IE 6 以下版本中左边界值*/
    position: relative;          /*相对定位，定义包含块*/
    }
.menu table {/*定义表格属性*/
    position: absolute;          /*绝对定位，精确确定子菜单项位置*/
    border-collapse: collapse;   /*合并单元格相邻边*/
    top: 0;                      /*精确定位坐标值*/
    left: 0;                     /*精确定位坐标值*/
    z-index: 100;                /*层叠顺序，在上面显示*/
    font-size: 1em;
    width: 0;                    /*定位宽为 0，实现隐藏*/
    height: 0;                   /*定位高为 0，实现隐藏*/
    }
.menu a, .menu a:visited {/*定义链接属性*/
    display: block;              /*块状显示，便于大小控制 */
    text-decoration: none;       /*清除下划线*/
    height: 25px;                /*定义超链接的高度*/
    line-height: 25px;           /*垂直居中显示*/
    width: 100px;                /*定义超链接的宽度*/
    color:#333;                  /*定义链接字体颜色*/
    text-indent: 5px;            /*缩进字体，使左侧留出空隙*/
    border-bottom: 1px solid #fff; /*定义超链接底部边框，实现菜单项间显示白色分割线*/
    background: #AFDD22;         /*定义菜单项背景色*/
    }
* html .menu a:hover {/*定义在 IE 7 以下浏览器中鼠标指针经过链接的样式*/
    color: #fff;
    background: #40DE5A;
    }
.menu :hover > a {/*定义在现代标准浏览器中链接样式*/
    color: #fff;
    background: #40DE5A;
    }
.menu ul ul {/*隐藏二级菜单，用绝对定位定义菜单，目的是不让其占据页面上的任何空间*/
    visibility: hidden;
    position: absolute;
    top: -1px;
    left: 100px;
```






```
    }
.menu ul li:hover ul, .menu ul a:hover ul {/*定义鼠标指针经过一级菜单时，显示对应的二级菜单*/
    visibility: visible;
    }
.menu ul :hover ul ul {/*定义当鼠标指针移动至一级菜单时，隐藏三级菜单*/
    visibility: hidden;
    }
.menu ul :hover ul :hover ul ul {/*定义当鼠标指针移动至二级菜单时，隐藏四级菜单*/
    visibility: hidden;
}
.menu ul :hover ul :hover ul {/*定义当鼠标指针经过二级菜单选项时，显示相对应的三级菜单*/
    visibility: visible;
    }
.menu ul :hover ul :hover ul :hover ul {/*定义鼠标指针经过三级菜单时，显示相对应的四级菜单*/
    visibility: visible;
    }
```

第 4 步，保存文档，按 F12 功能键，在浏览器中预览，效果如图 6.22 所示。

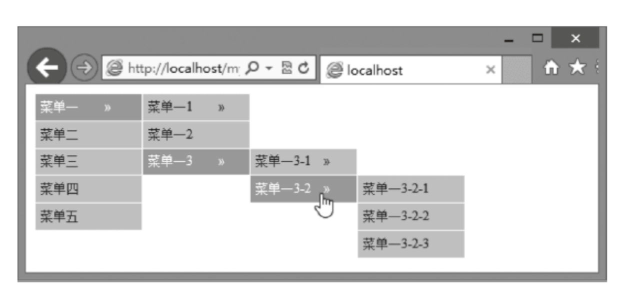


图 6.22　设计垂直多级菜单样式

6.3　在 线 练 习

下面通过大量的线上示例，帮助初学者练习使用 HTML5 设计表单结构和样式。感兴趣的读者可以扫码练习。



在 线 练 习1



在 线 练 习2



使用 CSS 美化表格

在传统网页设计中，表格的主要功能就是页面布局，因此表格是网页编辑的工具；在标准化网页设计中，表格的主要功能是显示数据，也可适当辅助结构设计。本章将详细介绍表格在网页设计中的应用，包括设计符合标准化的表格结构、正确设置表格属性，以及应用 CSS 表格样式。

【学习要点】

- 正确使用表格标签。
- 设置表格和单元格属性。
- 设计表格的 CSS 样式。




7.1 设 置 属 性


Note


视 频 讲 解

本节将介绍 table 和 td 的标签属性。

7.1.1 设置表格属性

表格标签包含大量属性，其中大部分属性都可以使用 CSS 属性代替，也有几个专用属性无法使用 CSS 实现。HTML5 支持的<table>标签属性及其说明如表 7.1 所示。

表 7.1 HTML5 支持的<table>标签属性及其说明

属　性	说　明
border	定义表格边框，值为整数，单位为像素。当值为 0 时，表示隐藏表格边框线。功能类似 CSS 中的 border 属性，但是没有 CSS 提供的边框属性强大
cellpadding	定义数据表单元格的补白。功能类似 CSS 中的 padding 属性，但是功能比较弱
cellspacing	定义数据表单元格的边界。功能类似 CSS 中的 margin 属性，但是功能比较弱
width	定义数据表的宽度。功能类似 CSS 中的 width 属性
frame	设置数据表的外边框线显示，实际上它是对 border 属性的功能扩展 取值包括 void（不显示任一边框线）、above（顶端边框线）、below（底部边框线）、hsides（顶部和底部边框线）、lhs（左边框线）、rhs（右边框线）、vsides（左和右的边框线）、box（四周的边框线）、border（四周的边框线）
rules	设置数据表的内边线显示，实际上它是对 border 属性的功能扩展。 取值包括 none（禁止显示内边线）、groups（仅显示分组内边线）、rows（显示每行的水平线）、cols（显示每列的垂直线）、all（显示所有行和列的内边线）
summary	定义表格的摘要，没有 CSS 对应属性

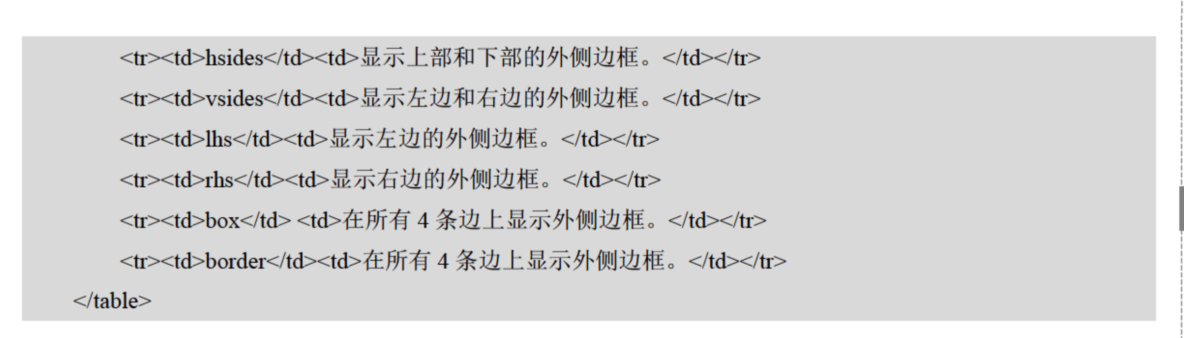
rules 和 frame 是两个特殊的表格样式属性，用于定义表格的各个内、外边框线是否显示。由于使用 CSS 的 border 属性可以实现相同的效果，所以不建议用户选用 rules 和 frame。这两个属性的取值可以参考表 7.1 中的说明。

【示例 1】本示例借助表格标签的 frame 和 rules 属性定义表格以单行线的形式进行显示。

```
<table border="1" frame="hsides"   rules="rows" width="100%">
    <caption>frame 属性取值说明</caption>
    <tr><th>值</th><th>说明</th></tr>
    <tr><td>void</td><td>不显示外侧边框。</td></tr>
    <tr><td>above</td><td>显示上部的外侧边框。</td></tr>
    <tr><td>below</td><td>显示下部的外侧边框。</td> </tr>
```

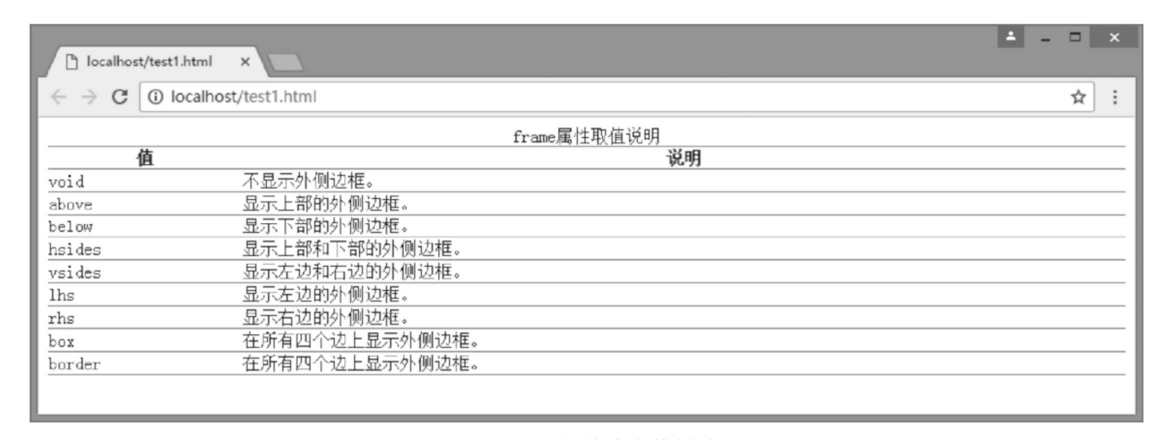






```
        <tr><td>hsides</td><td>显示上部和下部的外侧边框。</td></tr>
        <tr><td>vsides</td><td>显示左边和右边的外侧边框。</td></tr>
        <tr><td>lhs</td><td>显示左边的外侧边框。</td></tr>
        <tr><td>rhs</td><td>显示右边的外侧边框。</td></tr>
        <tr><td>box</td> <td>在所有 4 条边上显示外侧边框。</td></tr>
        <tr><td>border</td><td>在所有 4 条边上显示外侧边框。</td></tr>
</table>
```

上面示例通过 frame 属性定义表格仅显示上下框线，使用 rules 属性定义表格仅显示水平内边线，从而设计出单行线数据表格效果。在使用 frame 和 rules 属性时，同时定义 border 属性，指定数据表显示边框线。在浏览器中预览，则显示效果如图 7.1 所示。



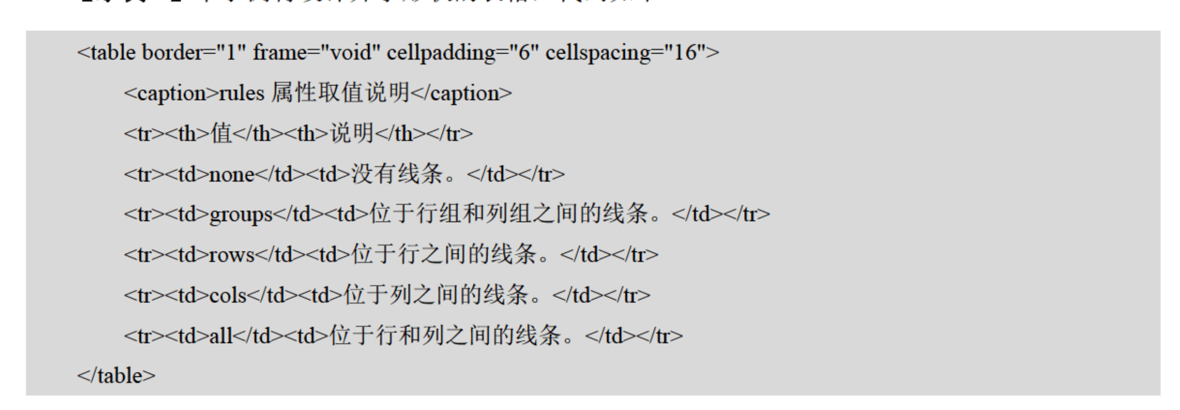
frame属性取值说明

值	说明
void	不显示外侧边框。
above	显示上部的外侧边框。
below	显示下部的外侧边框。
hsides	显示上部和下部的外侧边框。
vsides	显示左边和右边的外侧边框。
lhs	显示左边的外侧边框。
rhs	显示右边的外侧边框。
box	在所有四个边上显示外侧边框。
border	在所有四个边上显示外侧边框。

图 7.1　定义单线表格样式

cellpadding 属性用于定义单元格边沿与其内容之间的空白，cellspacing 属性定义单元格之间的空间。这两个属性的取值单位为像素或者百分比。

【示例 2】本示例将设计井字形状的表格，代码如下。



```
<table border="1" frame="void" cellpadding="6" cellspacing="16">
        <caption>rules 属性取值说明</caption>
        <tr><th>值</th><th>说明</th></tr>
        <tr><td>none</td><td>没有线条。</td></tr>
        <tr><td>groups</td><td>位于行组和列组之间的线条。</td></tr>
        <tr><td>rows</td><td>位于行之间的线条。</td></tr>
        <tr><td>cols</td><td>位于列之间的线条。</td></tr>
        <tr><td>all</td><td>位于行和列之间的线条。</td></tr>
</table>
```

上面示例通过 frame 属性隐藏表格外框，然后使用 cellpadding 属性定义单元格内容的边距为 6 像素，单元格之间的间距为 16 像素，则在浏览器中预览效果如图 7.2 所示。




Note



图 7.2　定义井字表格样式

提示：cellpadding 属性定义的效果，可以使用 CSS 的 padding 样式属性代替，不建议使用 cellpadding 属性。


视频讲解

7.1.2　设置单元格属性

单元格标签（<td>和<th>）包含大量属性，其中大部分属性都可以使用 CSS 属性代替，也有几个专用属性无法使用 CSS 实现。HTML5 支持的<td>和<th>标签属性及其说明如表 7.2 所示。

表 7.2　HTML5 支持的<td>和<th>标签属性及其说明

属　性	说　明
abbr	定义单元格中内容的缩写版本
align	定义单元格内容的水平对齐方式。取值包括 right（右对齐）、left（左对齐）、center（居中对齐）、justify（两端对齐）和 char（对准指定字符）。功能类似 CSS 中的 text-align 属性，建议使用 CSS 完成设计
axis	对单元进行分类。取值为一个类名
char	定义根据哪个字符来进行内容的对齐
charoff	定义对齐字符的偏移量
colspan	定义单元格可横跨的列数
headers	定义与单元格相关的表头
rowspan	定义单元格可横跨的行数
scope	定义将表头数据与单元格数据相关联的方法。取值包括 col（列的表头）、colgroup（列组的表头）、row（行的表头）、rowgroup（行组的表头）
valign	定义单元格内容的垂直排列方式。取值包括 top（顶部对齐）、middle（居中对齐）、bottom（底部对齐）、baseline（基线对齐）。功能类似 CSS 中的 vertical-align 属性，建议使用 CSS 完成设计





colspan 和 rowspan 是两个重要的单元格属性，分别用来定义单元格可跨列或跨行显示，取值为正整数。如果取值为 0，则表示浏览器横跨到列组的最后一列或者行组的最后一行。

【示例】 本示例使用 colspan=5 属性定义单元格跨列显示，代码如下，效果如图 7.3 所示。

```
<table border=1>
    <tr><th align=center colspan=5>课程表</th></tr>
    <tr><th>星期一</th><th>星期二</th> <th>星期三</th><th>星期四</th><th>星期五</th></tr>
    <tr><td align=center colspan=5>上午</td> </tr>
    <tr><td>语文</td><td>物理</td> <td>数学</td> <td>语文</td><td>美术</td></tr>
    <tr><td>数学</td><td>语文</td><td>体育</td> <td>英语</td><td>音乐</td></tr>
    <tr><td>语文</td> <td>体育</td><td>数学</td><td>英语</td><td>地理</td></tr>
    <tr><td>地理</td><td>化学</td><td>语文</td> <td>语文</td><td>美术</td></tr>
    <tr><td align=center colspan=5>下午</td></tr>
    <tr><td>作文</td><td>语文</td><td>数学</td><td>体育</td><td>化学</td></tr>
    <tr> <td>生物</td><td>语文</td><td>物理</td><td>自修</td><td>自修</td></tr>
</table>
```



课程表				
星期一	星期二	星期三	星期四	星期五
上午				
语文	物理	数学	语文	美术
数学	语文	体育	英语	音乐
语文	体育	数学	英语	地理
地理	化学	语文	语文	美术
下午				
作文	语文	数学	体育	化学
生物	语文	物理	自修	自修

图 7.3　定义单元格跨列显示

7.2　表格基本样式

CSS 为表格定义了 5 个专用属性，详细说明如表 7.3 所示。

表 7.3　CSS 表格属性列表

属　性	取　值	说　明
border-collapse	separate（边分开）| collapse（边合并）	定义表格的行和单元格的边是合并在一起还是按照标准的 HTML 样式分开
border-spacing	length	定义当表格边框独立（如当 border-collapse 属性等于 separate）时，行和单元格的边在横向和纵向上的间距，该值不可以取负值





续表

属　性	取　值	说　明
caption-side	top | bottom	定义表格的 caption 对象位于表格的顶部或底部。应与 caption 元素一起使用
empty-cells	show | hide	定义当单元格无内容时，是否显示该单元格的边框
table-layout	auto | fixed	定义表格的布局算法，可以通过该属性改善表格呈递性能，如果设置 fixed 属性值，会使 IE 以一次一行的方式呈递表格内容，从而提供给信息用户更快的速度；如果设置 auto 属性值，则表格在每一单元格内所有内容读取计算之后才会显示出来

除了表 7.3 介绍的 5 个表格专用属性外，CSS 其他属性对于表格一样适用。


视 频 讲 解

7.2.1　设计表格边框线

使用 CSS 的 border 属性代替<table>标签的 border 属性定义表格边框，可以优化代码结构。

【示例】本示例演示了如何使用 CSS 设计细线边框样式的表格。

第 1 步，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表。

第 2 步，在内部样式表中输入下面样式代码，定义单元格边框显示为 1 像素的灰色实线。

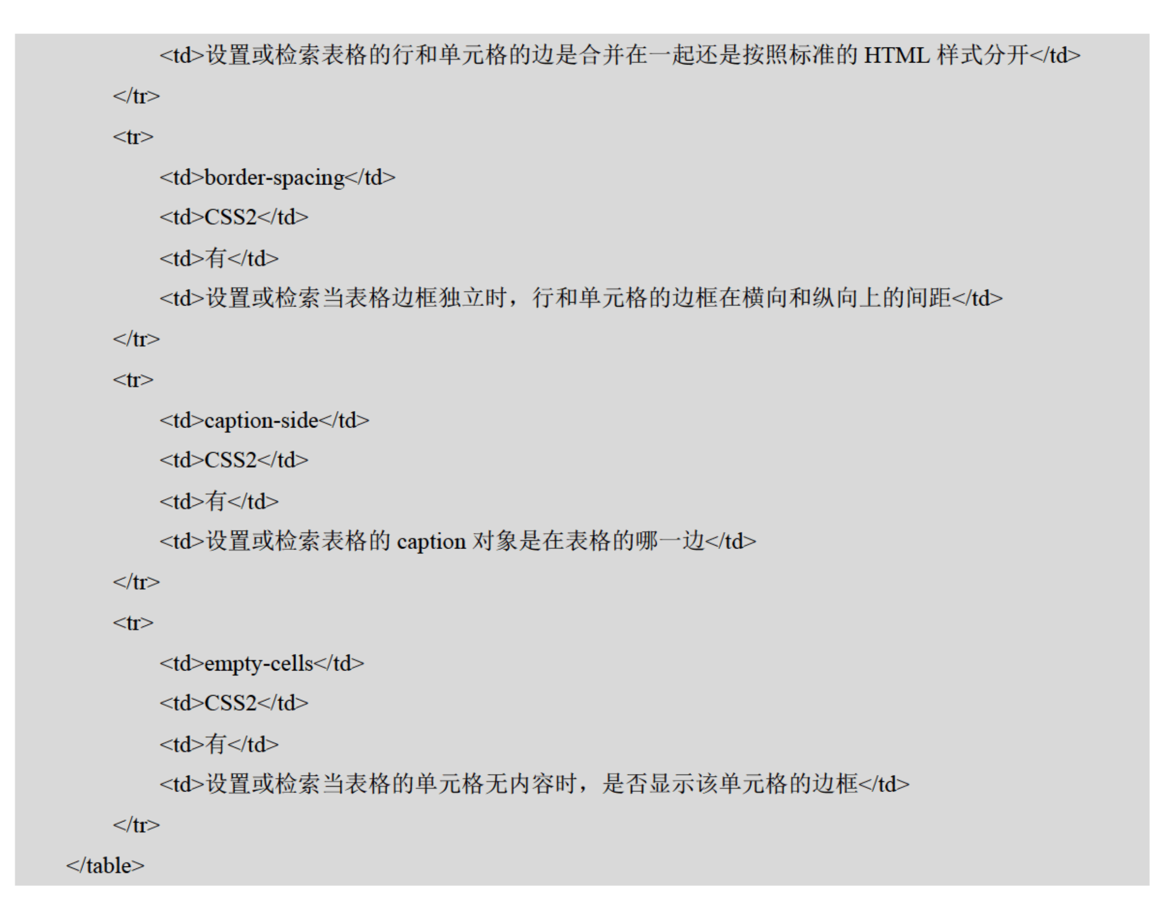
```
th, td {font-size:12px; border:solid 1px gray;}
```

第 3 步，在<body>标签内构建一个简单的表格结构，代码如下。

```
<table>
    <tr>
        <th>属性</th>
        <th>版本</th>
        <th>继承性</th>
        <th>描述</th>
    </tr>
    <tr>
        <td>table-layout</td>
        <td>CSS2</td>
        <td>无</td>
        <td>设置或检索表格的布局算法</td>
    </tr>
    <tr>
        <td>border-collapse</td>
        <td>CSS2</td>
        <td>有</td>
```







```
            <td>设置或检索表格的行和单元格的边是合并在一起还是按照标准的 HTML 样式分开</td>
        </tr>
        <tr>
            <td>border-spacing</td>
            <td>CSS2</td>
            <td>有</td>
            <td>设置或检索当表格边框独立时，行和单元格的边框在横向和纵向上的间距</td>
        </tr>
        <tr>
            <td>caption-side</td>
            <td>CSS2</td>
            <td>有</td>
            <td>设置或检索表格的 caption 对象是在表格的哪一边</td>
        </tr>
        <tr>
            <td>empty-cells</td>
            <td>CSS2</td>
            <td>有</td>
            <td>设置或检索当表格的单元格无内容时，是否显示该单元格的边框</td>
        </tr>
    </table>
```

第 4 步，在浏览器中预览，则显示效果如图 7.4 所示。



属性	版本	继承性	描述
table-layout	CSS2	无	设置或检索表格的布局算法
border-collapse	CSS2	有	设置或检索表格的行和单元格的边是合并在一起还是按照标准的HTML样式分开
border-spacing	CSS2	有	设置或检索当表格边框独立时，行和单元格的边框在横向和纵向上的间距
caption-side	CSS2	有	设置或检索表格的caption对象是在表格的哪一边
empty-cells	CSS2	有	设置或检索当表格的单元格无内容时，是否显示该单元格的边框

图 7.4　使用 CSS 定义单元格边框样式

通过效果图可以看到，使用 CSS 定义的单行线不是连贯的线条。这是因为表格中每个单元格都是一个独立的空间，为它们定义边框线时，相互之间不是紧密连接在一起的。

第 5 步，在内部样式表中，为 table 元素添加如下 CSS 样式，把相邻单元格进行合并。

```
table { border-collapse:collapse;}/*  合并单元格边框  */
```

第 6 步，在浏览器中重新预览页面效果，如图 7.5 所示。





Note



属性	版本	继承性	描述
table-layout	CSS2	无	设置或检索表格的布局算法
border-collapse	CSS2	有	设置或检索表格的行和单元格的边是合并在一起还是按照标准的HTML样式分开
border-spacing	CSS2	有	设置或检索当表格边框独立时，行和单元格的边框在横向和纵向上的间距
caption-side	CSS2	有	设置或检索表格的caption对象是在表格的哪一边
empty-cells	CSS2	有	设置或检索当表格的单元格无内容时，是否显示该单元格的边框

图 7.5　使用 CSS 合并单元格边框



视频讲解

7.2.2　定义单元格间距和空隙

为了兼容<table>标签的 cellspacing 属性，CSS 定义了 border-spacing 属性，该属性能够分离单元格间距。取值包含 1 个或 2 个值。当定义 1 个值时，则定义单元格行间距和列间距都为该值，例如：

```
table { border-spacing:20px;}/* 分隔单元格边框 */
```

如果分别定义行间距和列间距，就需要定义 2 个值，例如：

```
table { border-spacing:10px 30px;}/* 分隔单元格边框 */
```

其中第 1 个值表示单元格之间的行间距，第 2 个值表示单元格之间的列间距，该属性值不可以为负数。使用 cellspacing 属性定义单元格之间的距离之后，该空间由表格背景填充。

使用该属性应注意以下几个问题。

☑ 早期 IE 浏览器不支持该属性，要定义相同效果的样式，需要结合传统<table>标签的 cellspacing 属性来设置。

☑ 使用 cellspacing 属性时，应确保单元格之间相互的独立性，不能使用 border-collapse:collapse;样式定义合并表格内单元格的边框。

☑ cellspacing 属性不能够使用 CSS 的 margin 属性来代替。对于 td 元素来说，不支持 margin 属性。

☑ 可以为单元格定义补白，此时使用 CSS 的 padding 属性与单元格的 cellpadding 标签属性实现效果是相同的。

【示例 1】以 7.2.1 节示例中的表格结构为基础，重新设计内部样式表，为表格内单元格定义上下 6 像素和左右 12 像素的间距，同时设计单元格内部空隙为 12 像素。代码如下，演示效果如图 7.6 所示。

```
table { border-spacing: 6px 12px; }
th, td {
    font-size: 12px;
    border: solid 1px gray;
    padding: 12px;
}
```





属性	版本	继承性	描述
table-layout	CSS2	无	设置或检索表格的布局算法
border-collapse	CSS2	有	设置或检索表格的行和单元格的边是合并在一起还是按照标准的HTML样式分开
border-spacing	CSS2	有	设置或检索当表格边框独立时，行和单元格的边框在横向和纵向上的间距
caption-side	CSS2	有	设置或检索表格的caption对象是在表格的那一边
empty-cells	CSS2	有	设置或检索当表格的单元格无内容时，是否显示该单元格的边框

图 7.6　增加单元格空隙



也可以为<table>标签定义补白，此时可以增加表格外框与单元格之间的距离。

【示例 2】在示例 1 的基础上，为<table>标签重设如下样式，设计表格外框为 2 像素红色实线，定义表格外框与内部单元格间距为 2 像素。代码如下，显示效果如图 7.7 所示。

```
table {
    border-spacing: 6px 12px;
    border: solid 2px red;
    padding: 2px;
}
```



属性	版本	继承性	描述
table-layout	CSS2	无	设置或检索表格的布局算法
border-collapse	CSS2	有	设置或检索表格的行和单元格的边是合并在一起还是按照标准的HTML样式分开
border-spacing	CSS2	有	设置或检索当表格边框独立时，行和单元格的边框在横向和纵向上的间距
caption-side	CSS2	有	设置或检索表格的caption对象是在表格的那一边
empty-cells	CSS2	有	设置或检索当表格的单元格无内容时，是否显示该单元格的边框

图 7.7　为表格和单元格同时定义补白效果

7.2.3　隐藏空单元格



视 频 讲

如果表格单元格的边框处于分离状态（border-collapse: separate;），可以使用 CSS 的 empty-cells 属性设置空单元格是否显示。当其值为 show 时，表示显示空单元格；当值为 hide 时，表示隐藏空单元格。

【示例】在本示例中，隐藏第 2 行第 2 列的空单元格边框显示。代码如下，效果如图 7.8 所示。

```
<style type="text/css">
table {/* 表格样式 */
```




```
        width: 400px;                          /* 固定表格宽度 */
        border: dashed 1px red;                /* 定义虚线表格边框 */
        empty-cells: hide;                     /* 隐藏空单元格 */
    }
    th, td {/* 单元格样式 */
        border: solid 1px #000;                /* 定义实线单元格边框 */
        padding: 4px;                          /* 定义单元格内的补白区域 */
    }
    </style>

    <table>
        <tr><td>西</td><td>东</td> </tr>
        <tr><td>北</td><td></td></tr>
    </table>
```



隐藏空白单元格



默认显示的空白单元格

图 7.8 隐藏空单元格效果

> 提示：所谓空单元格，就是没有可视内容的单元格。如果单元格的 visibility 属性值为 hidden，即便单元格包含内容，也认为是无可视内容。而“ ”和其他空白字符为可视内容。ASCII 字符中的回车符（“\0D”）、换行符（“\0A”）、Tab 键（“\09”）和空格键（“\20”）表示无可视内容。

如果表格行中所有单元格的 empty-cells 属性都为 hide，且都不包含任何可视内容，那么整行就等于设置了 display: none。



视频讲解

7.2.4 定义标题样式

使用 CSS 的 caption-side 属性可以定义标题的显示位置，该属性取值包括 top（位于表格上面）、bottom（位于表格底部）、left（位于表格左侧，非标准）和 right（位于表格右侧，非标准）。

如果要水平对齐标题文本，则可以使用 text-align 属性。对于左右两侧的标题，可以使用 vertical-align 属性进行垂直对齐，取值包括 top、middle 和 bottom，其他取值无效，默认为 top。

【示例】在本示例中，定义标题靠左显示，并设置标题垂直居中显示。但不同浏览器在解析时分歧比较大，如在 IE 浏览器中显示效果如图 7.9 所示，但是在 Firefox 中的显示效果如图 7.10 所示。





```
<style type="text/css">
table {border: dashed 1px red; }                    /* 定义表格虚线外框样式 */
th, td {/* 定义单元格样式 */
    border: solid 1px #000;                         /* 实线内框 */
    padding: 20px 80px;                             /* 单元格内补白大小 */
}
caption {/* 定义标题行样式 */
    caption-side: left;                             /* 左侧显示 */
    width: 10px;                                    /* 定义宽度 */
    margin: auto 20px;                              /* 定义左右边界 */
    vertical-align: middle;                         /* 垂直居中显示 */
    font-size: 14px;                                /* 定义字体大小 */
    font-weight: bold;                              /* 加粗显示 */
    color: #666;                                    /* 灰色字体 */
}
</style>

<table>
    <caption>表格标题</caption>
    <tr><td>北</td><td>西</td> </tr>
    <tr><td>东</td><td>南</td> </tr>
</table>
```



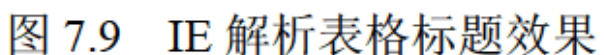


图 7.9　IE 解析表格标题效果



图 7.10　Firefox 解析表格标题效果

【拓展】

当同时为<table>、<tr>和<td>等标签定义背景色、边框、字体属性等样式时，就容易发生样式重叠问题。根据表格布局模型，各种表格对象背景样式层叠的顺序如图 7.11 所示。





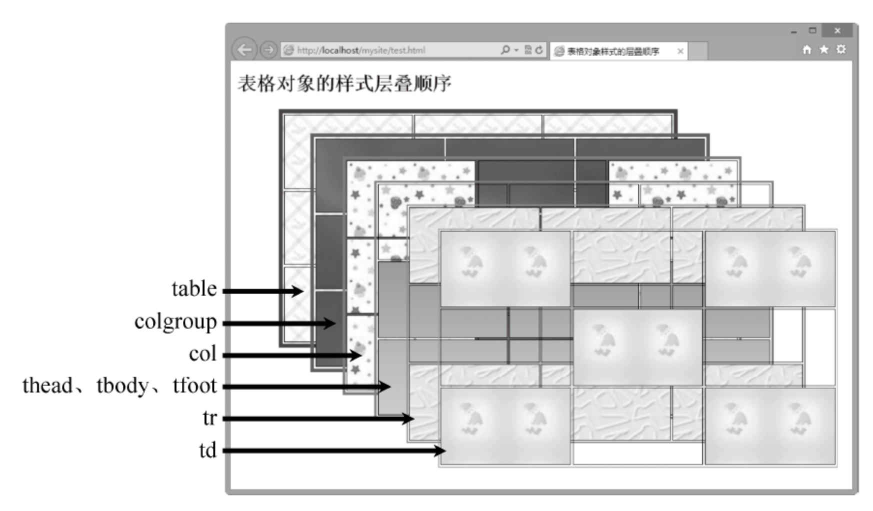


图 7.11　表格对象样式的层叠顺序

从图 7.11 可以看到：td 元素的样式具有最大优先权，以此类推，如果单元格为透明，则行（tr 元素）具有最大优先权。表格定义的背景优先权最小，当表格中其他元素都为透明时，才可以看到表格的背景。

7.3　案 例 实 战

本节将结合几个案例详细讲解表格样式的一般设计技巧。



7.3.1　设计斑马线表格

本例将对一个简单的表格进行设计，使它看起来更为精致。另外，当表格的行和列都很多，并且数据量很大的时候，为避免单元格采用相同的背景色会使浏览者感到凌乱，发生看错行的情况，为表格设置隔行变色的效果，使得奇数行和偶数行的背景色不一样，示例效果如图 7.12 所示。



历届奥运会中国奖牌数

编号	年份	城市	金牌	银牌	铜牌	总计
第23届	1984年	洛杉矶（美国）	15	8	9	32
第24届	1988年	汉城（韩国）	5	11	12	28
第25届	1992年	巴塞罗那（西班牙）	16	22	16	54
第26届	1996年	亚特兰大（美国）	16	22	12	50
第27届	2000年	悉尼（澳大利亚）	28	16	15	59
第28届	2004年	雅典（希腊）	32	17	14	63
第29届	2008年	北京（中国）	51	21	28	100
第30届	2012年	伦敦（英国）	38	27	23	88
第31届	2016年	里约热内卢（巴西）	26	18	26	70
合计		544枚				

图 7.12　设计隔行变色的样式效果





【操作步骤】

第 1 步，新建 HTML5 文档，设计表格结构，代码如下。

```
<table summary="历届奥运会中国奖牌数">
    <caption>历届奥运会中国奖牌数</caption>
        <tr><th>编号</th><th>年份</th><th>城市</th><th>金牌</th><th>银牌</th><th>铜牌</th><th>总计
</th></tr>
    </thead>
    <tbody>
        <tr><td>第 23 届</td><td>1984 年</td><td>洛杉矶（美国）</td><td>15</td><td>8</td><td>9</td>
<td>32</td></tr>
        <tr><td>第 24 届</td><td>1988 年</td><td>汉城（韩国）</td><td> 5</td><td>11</td><td>12</td>
<td>28</td></tr>
        <tr><td>第 25 届</td><td>1992 年</td><td>巴塞罗那（西班牙）</td><td>16</td><td>22</td><td>
16</td><td>54</td></tr>
        <tr><td>第 26 届</td><td>1996 年</td><td>亚特兰大（美国）</td><td>16</td><td>22</td><td>
12</td><td>50</td></tr>
        <tr><td>第 27 届</td><td>2000 年</td><td>悉尼（澳大利亚）</td><td>28</td><td>16</td><td>
15</td><td>59</td></tr>
        <tr><td>第 28 届</td><td>2004 年</td><td>雅典（希腊）</td><td>32</td><td>17</td><td>
14</td><td>63</td></tr>
        <tr><td>第 29 届</td><td>2008 年</td><td>北京（中国）</td><td>51</td><td>21</td><td>
28</td><td>100</td></tr>
        <tr><td>第 30 届</td><td>2012 年</td><td>伦敦（英国）</td><td>38</td><td>27</td><td>
23</td><td>88</td></tr>
        <tr><td>第 31 届</td><td>2016 年</td><td>里约热内卢（巴西）</td><td>26</td><td>18</td><td>
26</td><td>70</td></tr>
    </tbody>
    <tfoot>
        <tr><th>合计</th><td colspan="4">544 枚</td></tr>
    </tfoot>
</table>
```

在这个表格中，使用的标记从上至下依次为<caption>、<thead>、<tbody>和<tfoot>，分别定义表格的标题、列标题行、数据区域和脚注行。

第 2 步，在头部区域<head>标签中插入一个<style type="text/css">标签，在该标签中输入下面的样式代码，定义表格框和表格标题样式。

```
table {/*表格样式 */
```





Note

```
        background-color: #FFF; color: #565;        /* 表格背景色和字体颜色 */
        border: none;                               /* 清除表格外框线 */
        font: 12px arial;                           /* 设置字体大小和类型 */
        width: 90%;                                 /* 定义弹性宽度 */
        margin: 12px auto;                          /* 表格水平居中显示 */
}
table caption {/*表格标题样式 */
        font-size: 24px;                            /* 定义表格标题字体大小 */
        border-bottom: 2px solid #B3DE94;           /* 添加下边线 */
        border-top: 2px solid #B3DE94;              /* 添加上边线 */
}
```

第 3 步，设计数据行单元格样式，代码如下。

```
tbody td, tbody th {
        background-color: #DFC;                     /* 定义背景色 */
        border-bottom: 2px solid #B3DE94;           /* 添加下边线 */
        border-top: 3px solid #FFFFFF;              /* 添加上边线 */
        padding: 9px;                               /* 增大单元格空间 */
}
```

第 4 步，设计脚注行单元格样式，代码如下。

```
tfoot td, tfoot th {
        font-weight: bold;                          /* 加粗显示 */
        padding: 4px 8px 6px 9px;                   /* 增大单元格空间 */
        text-align: center;                         /* 居中显示 */
}
```

第 5 步，设计列标题行样式，代码如下。

```
thead th {
        font-size: 14px; line-height: 19px;
        padding: 0 8px 2px;
}
```

第 6 步，为最后 4 列单元格定义样式，代码如下。

```
tbody td+td+td+td{
        color:red;
        text-align:right;
}
```





这里将 3 个相邻选择器组合使用，精确匹配到后面 4 列单元格标签，然后设置字体颜色为红色，文本右对齐显示。

【拓展】

用户可以使用<col>或<colgroup>标签为各列定义样式。例如，下面代码中第 2 个<col span="4">标签可以为最后 4 列定义样式。

```
<table summary="历届奥运会中国奖牌数">
    <caption>
    历届奥运会中国奖牌数
    </caption>
    <col span="3"></col>
    <col span="4" style="color:red; text-align:right"></col>
    <thead>
    …
</table>
```

但是，现代标准浏览器仅支持 background-color 和 width 属性，不支持其他 CSS 属性。IE 诡异模式支持所有 CSS 属性。

第 7 步，设计隔行换色样式。隔行换色是一款比较经典的表格样式，这种样式主要是从用户体验角度来设计的，以提升用户浏览数据的速度和准确度。隔行换色的传统设计方法是：定义一个类，然后把该类应用到所有奇数行或偶数行。推荐方法：使用 CSS3 选择器匹配表格中偶数行和奇数行，并设计不同的样式，代码如下。

```
tbody tr:nth-child(odd) td {
    background-color: #CEA;
    border-bottom: 2px solid #67BD2A;
}
```

第 8 步，设计鼠标指针移过数据行时的交互样式，代码如下。

```
tbody tr:hover td, tbody tr:hover th {
    background-color: #8b7;
    color: #fff;
}
```

7.3.2 设计粗线框表格



本例以 7.3.1 节案例的数据表格结构为基础，介绍如何使用 CSS 把表格设计为网格化效果，以此理解 CSS 控制表格的方法，案例效果如图 7.13 所示。







历届奥运会中国奖牌数

编号	年份	城市	金牌	银牌	铜牌	总计
第23届	1984年	洛杉矶（美国）	15	8	9	32
第24届	1988年	汉城（韩国）	5	11	12	28
第25届	1992年	巴塞罗那（西班牙）	16	22	16	54
第26届	1996年	亚特兰大（美国）	16	22	12	50
第27届	2000年	悉尼（澳大利亚）	28	16	15	59
第28届	2004年	雅典（希腊）	32	17	14	63
第29届	2008年	北京（中国）	51	21	28	100
第30届	2012年	伦敦（英国）	38	27	23	88
第31届	2016年	里约热内卢（巴西）	26	18	26	70
合计	544枚					

图 7.13　设计粗线框表格

【操作步骤】

第 1 步，新建 HTML5 文档，复制 7.3.1 节示例的数据表格结构，把第 1 列单元格<td>标签改为<th>标签。

第 2 步，规划整个页面的基本显示属性并设置表格样式，代码如下。

```
body { /*网页基本样式类*/
    background-color: #f8e6e6;          /*网页背景颜色*/
    margin: 50px;                       /*表格四周补白*/
}
table {/*表格样式*/
    border: 6px double #3186dd;         /*  表格边框  */
    font-family: Arial;
    text-align: center;                 /*表格中文字水平居中对齐*/
    border-collapse: collapse;          /*  边框重叠  */
}
```

此时显示效果如图 7.14 所示。可以看到，网页背景颜色发生了改变，且表格显示了边框。

第 3 步，设置表格标题的样式，代码如下。

```
table caption   {                       /*设置表格标题  */
    padding-top: 3px;                   /*设置表格标题的顶部边距  */
    padding-bottom: 4px;                /*设置表格标题的底部边距  */
    font-size: 30px;                    /*表格标题字体大小  */
    color: red;                         /*表格标题字体颜色  */
}
```







历届奥运会中国奖牌数

编号	年份	城市	金牌	银牌	铜牌	总计
第23届	1984年	洛杉矶（美国）	15	8	9	32
第24届	1988年	汉城（韩国）	5	11	12	28
第25届	1992年	巴塞罗那（西班牙）	16	22	16	54
第26届	1996年	亚特兰大（美国）	16	22	12	50
第27届	2000年	悉尼（澳大利亚）	28	16	15	59
第28届	2004年	雅典（希腊）	32	17	14	63
第29届	2008年	北京（中国）	51	21	28	100
第30届	2012年	伦敦（英国）	38	27	23	88
第31届	2016年	里约热内卢（巴西）	26	18	26	70
合计	544枚					

图 7.14　设置网页基本属性及表格样式

第 4 步，设置表格中的单元格样式，代码如下。

```
table th {/* 表格的行、列名称单元格的样式 */
    border: 2px solid #429fff;          /* 行、列名称边框 */
    background-color: #d2e8ff;          /* 行、列名称单元格的背景颜色 */
    font-weight: bold;                  /* 行、列名称字体加粗 */
    padding-top: 4px;                   /* 设置行、列名称单元格的上、下、左、右边距 */
    padding-bottom: 4px;
    padding-left: 10px;
    padding-right: 10px;
}
table td {/* 表格单元格样式 */
    border: 2px solid #429fff;          /* 单元格边框 */
}
```

在以上代码中，分别设置了<th>和<td>标签的样式，并对表格的单元格进行了背景颜色、边框样式设置，从而达到美化表格的目的。至此，整个案例设计完成。

7.3.3 设计浅色风格表格



本例通过浅色搭配设计一款淡雅的表格效果，同时设置表格隔行变色，使奇数行和偶数行背景颜色不同，让数据行看起来清晰、明了，示例效果如图 7.15 所示。







历届奥运会中国奖牌数

编号	年份	城市	金牌	银牌	铜牌	总计
第23届	1984年	洛杉矶（美国）	15	8	9	32
第24届	1988年	汉城（韩国）	5	11	12	28
第25届	1992年	巴塞罗那（西班牙）	16	22	16	54
第26届	1996年	亚特兰大（美国）	16	22	12	50
第27届	2000年	悉尼（澳大利亚）	28	16	15	59
第28届	2004年	雅典（希腊）	32	17	14	63
第29届	2008年	北京（中国）	51	21	28	100
第30届	2012年	伦敦（英国）	38	27	23	88
第31届	2016年	里约热内卢（巴西）	26	18	26	70
合计	544枚					

图 7.15　设计浅色风格表格

【操作步骤】

第 1 步，新建 HTML5 文档，复制 7.3.2 节示例的数据表格结构。

第 2 步，定义网页基本属性、表格<table id="mytable">的样式，以及表格标题样式，代码如下。

```
body { /*网页基本样式*/
    background: #E6EAE9;
}
table { /*表格样式*/
    width: 700px;                        /*表格宽度*/
    padding: 0;
    margin: 0;
    border: 1px solid #C1DAD7;           /*表格边框*/
    border-collapse: collapse;
}
caption { /*设置表格标题 */
    padding: 0 0 5px 0;
    text-align: center;                  /*水平居中*/
    font-size: 30px;                     /*字体大小*/
    font-weight: bold;                   /*字体加粗*/
}
```

在以上代码中，首先定义了页面的背景颜色，在#mytable 中设置了表格的宽度为 700px，并为其添加了表格边框，此时的显示效果如图 7.16 所示。为 table 添加 border-collapse: collapse;声明，以解决单元格边框分离问题。







编号	年份	城市	金牌	银牌	铜牌	总计
第23届	1984年	洛杉矶（美国）	15	8	9	32
第24届	1988年	汉城（韩国）	5	11	12	28
第25届	1992年	巴塞罗那（西班牙）	16	22	16	54
第26届	1996年	亚特兰大（美国）	16	22	12	50
第27届	2000年	悉尼（澳大利亚）	28	16	15	59
第28届	2004年	雅典（希腊）	32	17	14	63
第29届	2008年	北京（中国）	51	21	28	100
第30届	2012年	伦敦（英国）	38	27	23	88
第31届	2016年	里约热内卢（巴西）	26	18	26	70
合计	544枚					

图 7.16　设置表格基本属性

第 3 步，使用 thead th 选择器单独为列标题行定义样式，使用 tbody 选择器定义数据区域背景色，代码如下。

```
thead th {
    color: #4f6b72;
    border: 1px solid #C1DAD7;
    letter-spacing: 2px;
    text-align: left;
    padding: 6px 6px 6px 12px;
    background: #CAE8EA;
}
tbody { background: #fff;}
```

第 4 步，使用 tbody th, tbody td 组合选择器，为数据区单元格定义样式，这样就避免了为每个单元格引用类样式，代码如下。

```
tbody th, tbody td {
    border: 1px solid #C1DAD7;
    font-size: 14px;
    padding: 6px 6px 6px 12px;
    color: #4f6b72;
}
```

第 5 步，使用 CSS3 的结构伪类选择器 tbody tr:nth-child(2n)专门为数据区域内所有偶数行定义特殊样式，实现隔行换色效果，这样避免了单独为偶数行单元格应用特殊类样式，代码如下。

```
tbody tr:nth-child(2n) {
    background: #F5FAFA;
    color: #797268;
}
```



7.3.4 设计清新风格表格



Note



视频讲解

本例设计这样一款表格样式：整体色调以清淡为主，边框线以淡蓝色为主色调，并配以 12 像素的灰色字体，营造一种轻松的视觉效果。然后，使用隔行换色样式分行显示数据，这也是目前数据表格的主流样式，它符合视线的换行显示，避免错行阅读数据。使用渐变背景图像来设计表格列标题，使表格看起来更大方，富有立体感。

【操作步骤】

第 1 步，新建 HTML5 文档，复制 7.3.3 节示例的数据表格结构。

第 2 步，定义表格样式。表格样式包括 3 部分内容：表格边框和背景样式、表格内容显示样式和表格布局样式。布局样式包括：定义表格固定宽度解析，这样能够优化解析速度；显示空单元格；合并单元格的边框线；设置表格居中显示。同时设置表格边框为 1 像素宽的浅蓝色实线框，字体大小固定为 12 像素的灰色字体，代码如下。

```
table {/* 表格基本样式 */
    table-layout:fixed;                /* 固定表格布局，优化解析速度*/
    empty-cells:show;                  /* 显示空单元格 */
    margin:0 auto;                     /* 居中显示 */
    border-collapse: collapse;         /* 合并单元格边框 */
    border:1px solid #cad9ea;          /* 边框样式 */
    color:#666;                        /* 灰色字体 */
    font-size:12px;                    /* 字体大小 */
}
```

提示：table-layout 是 CSS 定义的一个标准属性，用来设置表格布局的算法，取值包括 auto 和 fixed。当取值为 auto 时，则布局将基于单元格内包含的内容来进行布局，表格在每一单元格内所有内容读取计算之后才会显示出来；当取值为 fixed 时，表示固定布局算法，在这种算法中，表格和列的宽度取决于 col 对象的宽度总和，如果没有指定，则根据第一行每个单元格的宽度。如果表格没有指定宽度，则表格被呈递的默认宽度为 100%。设置 auto 布局算法，需要进行两次布局计算，这会影响客户端的解析速度，而 fixed 布局算法仅需要计算一次，所以速度非常快。

第 3 步，定义列标题样式。列标题样式主要涉及背景图像的设计，具体代码如下。

```
th {/* 列标题样式 */
    background-image: url(images/th_bg1.gif);      /* 指定渐变背景图像 */
    background-repeat:repeat-x;                    /* 定义水平平铺 */
    height:30px;                                   /* 固定高度 */
}
```





列标题样式的设计难点是背景图像的制作，具体制作方法这里就不再详细讲解，读者可以参考本示例效果，在 Photoshop 中进行设计。

第 4 步，定义单元格的显示样式。这里主要定义单元格的高度、边框线和补白。定义单元格左右两侧补白的目的是避免单元格与数据拥挤在一起，代码如下。

```
td {height:20px; /* 固定高度 */}              /* 单元格的高度 */
td, th {/* 单元格的边框线和补白 */
    border:1px solid #cad9ea;                 /* 单元格边框线应与表格边框线一致 */
    padding:0 1em 0;                          /* 单元格左右两侧的补白，一个字距 */
}
```

第 5 步，定义隔行变色样式，定义比边框色稍浅的背景色，代码如下。

```
tbody tr:nth-child(2n) { background-color: #f5fafe;}
```

第 6 步，保存页面，在浏览器中预览，显示效果如图 7.17 所示。

历届奥运会中国奖牌数

编号	年份	城市	金牌	银牌	铜牌	总计
第23届	1984年	洛杉矶（美国）	15	8	9	32
第24届	1988年	汉城（韩国）	5	11	12	28
第25届	1992年	巴塞罗那（西班牙）	16	22	16	54
第26届	1996年	亚特兰大（美国）	16	22	12	50
第27届	2000年	悉尼（澳大利亚）	28	16	15	59
第28届	2004年	雅典（希腊）	32	17	14	63
第29届	2008年	北京（中国）	51	21	28	100
第30届	2012年	伦敦（英国）	38	27	23	88
第31届	2016年	里约热内卢（巴西）	26	18	26	70
合计	544枚					

图 7.17　设计表格效果

7.3.5　设计圆润边框表格



本例使用 CSS3 新技术设计圆润风格的表格效果：使用 border-radius 定义圆角；使用 box-shadow 为表格添加内阴影，设计高亮边效果；使用 transition 定义过渡动画，让鼠标指针移过数据行，渐显浅色背景；使用 linear-gradient()函数定义标题列渐变背景效果，以替换传统使用背景图像模拟的渐变效果；使用 text-shadow 属性定义文本阴影，让标题文本看起来更富立体感。演示效果如图 7.18 所示。

【操作步骤】

第 1 步，新建 HTML5 文档，复制 7.3.4 节示例的数据表格结构。

第 2 步，在头部区域<head>标签中插入一个<style type="text/css">标签，在该标签中输入下面样式代码，定义表格默认样式，并定制表格外框主题类样式。

```
table {
    *border-collapse: collapse; /*兼容 IE 7 及以下版本浏览器 */
```




```
        border-spacing: 0;
        width: 100%;
        }
    .bordered {
        border: solid #ccc 1px;
        border-radius: 6px;
        box-shadow: 0 1px 1px #ccc;
        }
```



历届奥运会中国奖牌数

编号	年份	城市	金牌	银牌	铜牌	总计
第23届	1984年	洛杉矶（美国）	15	8	9	32
第24届	1988年	汉城（韩国）	5	11	12	28
第25届	1992年	巴塞罗那（西班牙）	16	22	16	54
第26届	1996年	亚特兰大（美国）	16	22	12	50
第27届	2000年	悉尼（澳大利亚）	28	16	15	59
第28届	2004年	雅典（希腊）	32	17	14	63
第29届	2008年	北京（中国）	51	21	28	100
第30届	2012年	伦敦（英国）	38	27	23	88
第31届	2016年	里约热内卢（巴西）	26	18	26	70
合计	544枚					

图 7.18　设计圆润边框表格

第 3 步，继续输入下面样式，统一单元格样式，并定义边框、空隙效果。

```
    .bordered td,  .bordered th {
        border-left: 1px solid #ccc;
        border-top: 1px solid #ccc;
        padding: 10px;
        text-align: left;
        }
```

第 4 步，输入下面样式代码，设计表格标题列样式，通过渐变效果设计标题列背景效果，并适当添加阴影，营造立体效果。

```
    .bordered th {
        background-color: #dce9f9;
        background-image: linear-gradient(top, #ebf3fc, #dce9f9);
        box-shadow: 0 1px 0 rgba(255,255,255,.8) inset;
        border-top: none;
        text-shadow: 0 1px 0 rgba(255,255,255,.5);
        }
```





第 5 步，输入下面样式代码，设计圆角效果。在制作表格圆角效果之前，有必要先完成这一步。表格的 border-collapse 默认值是 separate，将其值设置为 0，也就是 border-spacing:0;。

```
table {
    *border-collapse: collapse; /*兼容 IE7 及以下版本浏览器 */
    border-spacing: 0;
}
```

为了能兼容 IE 7 以及更低版本的浏览器，需要加上一个特殊的属性：border-collapse，并且将其值设置为 collapse。

第 6 步，设计圆角效果，具体代码如下。

```
/*==整个表格设置了边框，并设置了圆角==*/
.bordered { border: solid #ccc 1px; border-radius: 6px;}
/*==表格头部第一个 th 需要设置一个左上角圆角==*/
.bordered th:first-child { border-radius: 6px 0 0 0;}
/*==表格头部最后一个 th 需要设置一个右上角圆角==*/
.bordered th:last-child { border-radius: 0 6px 0 0;}
/*==表格最后一行的第一个 td 需要设置一个左下角圆角==*/
.bordered tr:last-child td:first-child {border-radius: 0 0 0 6px;}
/*==表格最后一行的最后一个 td 需要设置一个右下角圆角==*/
.bordered tr:last-child td:last-child {border-radius: 0 0 6px 0;}
```

第 7 步，由于在 table 中设置了一个边框，为了显示圆角效果，需要在表格的 4 个角的单元格上分别设置圆角效果，并且其圆角效果需要和表格的圆角值大小一样，反之，如果在 table 上没有设置边框，只需要在表格的 4 个角落的单元格设置圆角，就能实现圆角效果，代码如下。

```
/*==表格头部第一个 th 需要设置一个左上角圆角==*/
.bordered th:first-child { border-radius: 6px 0 0 0;}
/*==表格头部最后一个 th 需要设置一个右上角圆角==*/
.bordered th:last-child { border-radius: 0 6px 0 0;}
/*==表格最后一行的第一个 td 需要设置一个左下角圆角==*/
.bordered tfoot td:first-child {border-radius: 0 0 0 6px;}
/*==表格最后一行的最后一个 td 需要设置一个右下角圆角==*/
.bordered tfoot td:last-child {border-radius: 0 0 6px 0;}
```

在上面的代码中，使用了许多 CSS3 的伪类选择器。

第 8 步，除了使用 CSS3 选择器外，本案例还采用了很多 CSS3 的相关属性，这些属性将在后面章节中进行详细介绍，现简单概括如下。

使用 box-shadow 制作表格的阴影，代码如下。



```
.bordered { box-shadow: 0 1px 1px #ccc;}
```



Note

使用 transition 制作 hover 过渡效果，代码如下。

```
.bordered tr {transition: all 0.1s ease-in-out;}
```

使用 gradient 制作表头渐变色，代码如下。

```
.bordered th {
    background-color: #dce9f9;
    background-image: linear-gradient(to top, #ebf3fc, #dce9f9);
}
```

第 9 步，使用 CSS3 的 text-shadow 来制作文字阴影效果，rgba 改变颜色透明度等。

第 10 步，为<table>标签应用 bordered 类样式，代码如下。

```
<table summary="历届奥运会中国奖牌数"    class="bordered">
```



视频讲解

7.3.6　设计数据分组表格

本例通过树形结构来设计层次清晰的分类数据表格效果。整个表格样式设计包含以下 4 个技巧。

☑　适当修改数据表格的结构，使其更利于树形结构的设计。

☑　借助背景图像应用技巧来设计树形结构标志。

☑　借助伪类选择器来设计鼠标指针经过行时变换背景颜色。

☑　通过边框和背景色来设计列标题的立体显示效果。

【操作步骤】

第 1 步，新建 HTML5 文档，复制 7.3.5 节示例的数据表格结构。

第 2 步，修改数据表的结构。在修改数据表结构时，不要破坏数据表的基本结构，主要强化数据表格的分组。使用 thead 把标题分为一组（标题区域），使用多个 tbody 把数据分为多组（数据区域）。根据数据分类的需要，在每个 tbody 内部增加一个合并的数据行，该行仅包含了一个单元格，为了避免破坏结构，使用 colspan="7"合并单元格。经过修改之后的数据表格结构如下。

```
<table summary="历届奥运会中国奖牌数">
    <caption>历届奥运会中国奖牌数</caption>
    <thead>
        <tr></tr>
    </thead>
    <tbody>
      <tr><td colspan="7">第一时期</td></tr>
        <tr>……</tr>
        <tr>……</tr>
```





```
            <tr>……</tr>
            <tr>……</tr>
            <tr>……</tr>
            <tr>……</tr>
        </tbody>
        <tbody>
          <tr><td colspan="7">第二时期</td></tr>
            <tr>……</tr>
            <tr>……</tr>
            <tr>……</tr>
        </tbody>
        <tfoot>
            <tr><th>合计</th><td colspan="6">544 枚</td></tr>
        </tfoot>
</table>
```

第 3 步，重置基本表格对象的默认样式。例如，在 body 元素中定义页面字体类型，通过 table 元素定义数据表格的基本属性，以及其包含文本的基本显示样式。同时统一标题单元格和普通单元格的基本样式，代码如下。

```
body {font-family:"宋体" arial, helvetica, sans-serif; /* 页面字体类型 */}/* 页面基本属性 */
table {/* 表格基本样式 */
    border-collapse: collapse;              /* 合并单元格边框 */
    font-size: 85%;                         /* 字体大小，约为 14 像素 */
    line-height: 1.1;                       /* 行高，使数据显得更紧凑 */
    width: 96%;                             /* 固定宽度 */
    margin: auto;                           /* 水平居中显示 */
    border:solid 6px #c6ceda;               /* 添加粗边框，颜色与标题行背景色一致 */
}
th {/* 列标题基本样式 */
    font-weight: normal;                    /* 普通字体，不加粗显示 */
    text-align: left;                       /* 标题左对齐 */
    padding-left: 15px;                     /* 定义左侧补白 */
}
th, td {padding: .6em .6em; /* 增加补白效果，避免数据拥挤在一起 */}/* 单元格基本样式 */
```

第 4 步，定义列标题的立体效果。列标题的立体效果主要借助边框样式来实现，设计顶部、左侧和右侧边框样式为像素宽的白色实线，而底部边框则设计为 2 像素宽的浅灰色实线，这样就可以营造出一种淡淡的立体凸起效果，代码如下。




Note

```
thead th,tfoot th, tfoot td {/* 列标题样式，立体效果 */
    background: #c6ceda;                    /* 背景色 */
    border-color: #fff #fff #888 #fff;      /* 配置立体边框效果 */
    border-style: solid;                    /* 实线边框样式 */
    border-width: 1px 1px 2px 1px;          /* 定义边框大小 */
    padding-left: .5em;                     /* 增加左侧的补白 */
}
```

第 5 步，定义树形结构效果。树形结构主要利用虚线背景图像（ 和 ）来模拟，借助背景图像的灵活定位特性，可以精确设计出树形结构样式。然后使用结构伪类选择器分别把它们应用到每行的第一个单元格中，代码如下。

```
tbody tr td:first-child {/* 树形结构非末行图标样式 */
    background: url(images/dots.gif) 18px 54% no-repeat;     /* 定义树形结构末行图标 */
    padding-left: 26px;                                      /* 增加左侧的补白 */
}
tbody tr:last-child td:first-child {/* 树形结构末行图标样式 */
    background: url(images/dots2.gif) 18px 54% no-repeat;    /* 定义树形结构的末行图标 */
    padding-left: 26px;                                      /* 增加左侧的补白 */
}
```

第 6 步，为分类标题行定义一个样式类。通过为该行增加一个提示图标以及行背景色，来区分不同分类行之间的视觉分类效果，代码如下。

```
tbody tr:first-child td {/* 数据分类标题行的样式 */
    background:#eee url(images/arrow.gif) no-repeat 12px 50%;    /* 背景图像，定义提示图标 */
    padding-left: 28px;                                          /* 增加左侧的补白 */
    font-weight:bold;                                            /* 字体加粗显示 */
    color:#444;                                                  /* 字体颜色 */
}
```

第 7 步，设计当鼠标指针经过每行时变换背景色颜色，以此显示当前行效果，代码如下。

```
tr:hover, td.start:hover, td.end:hover {/* 鼠标指针经过行、单元格上时的样式 */
    background: #FF9;                       /* 变换背景色 */
}
```

第 8 步，保存页面，在浏览器中预览，显示效果如图 7.19 所示。







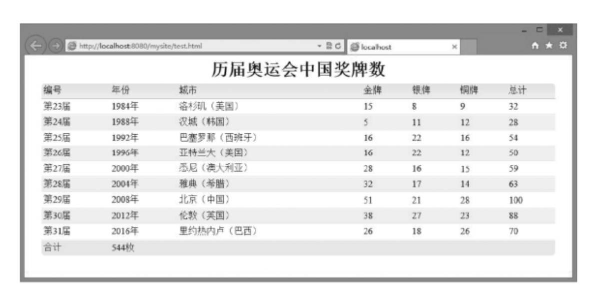
历届奥运会中国奖牌数

编号	年份	城市	金牌	银牌	铜牌	总计
第一时期						
第23届	1984年	洛杉矶（美国）	15	8	9	32
第24届	1988年	汉城（韩国）	5	11	12	28
第25届	1992年	巴塞罗那（西班牙）	16	22	16	54
第26届	1996年	亚特兰大（美国）	16	22	12	50
第27届	2000年	悉尼（澳大利亚）	28	16	15	59
第28届	2004年	雅典（希腊）	32	17	14	63
第二时期						
第29届	2008年	北京（中国）	51	21	28	100
第30届	2012年	伦敦（英国）	38	27	23	88
第31届	2016年	里约热内卢（巴西）	26	18	26	70
合计	544枚					

图 7.19　设计数据分组表格效果

7.3.7　设计单线表格

本例在前面案例的数据表格结构的基础上，使用 CSS3 技术设计一款单行线表格，效果如图 7.20 所示。



历届奥运会中国奖牌数

编号	年份	城市	金牌	银牌	铜牌	总计
第23届	1984年	洛杉矶（美国）	15	8	9	32
第24届	1988年	汉城（韩国）	5	11	12	28
第25届	1992年	巴塞罗那（西班牙）	16	22	16	54
第26届	1996年	亚特兰大（美国）	16	22	12	50
第27届	2000年	悉尼（澳大利亚）	28	16	15	59
第28届	2004年	雅典（希腊）	32	17	14	63
第29届	2008年	北京（中国）	51	21	28	100
第30届	2012年	伦敦（英国）	38	27	23	88
第31届	2016年	里约热内卢（巴西）	26	18	26	70
合计	544枚					

图 7.20　设计单线表格效果

【操作步骤】

第 1 步，新建 HTML5 文档，复制 7.3.6 节示例的数据表格结构。

第 2 步，在头部区域<head>标签中插入一个<style type="text/css">标签，在该标签中输入下面样式代码，定义表格默认样式，并定制表格外框主题类样式。

```
table {
    *border-collapse: collapse; /* IE7 and lower */
    border-spacing: 0;
    width: 100%;
}
```

第 3 步，设计单元格样式以及标题单元格样式，取消标题单元格的默认加粗和居中显示，代码如下。

```
.table td, .table th {
    padding: 4px;                       /* 增大单元格补白，避免拥挤 */
    border-bottom: 1px solid #f2f2f2;   /* 定义下边框线 */
    text-align: left;                   /* 文本左对齐 */
    font-weight:normal;                 /* 取消加粗显示 */
}
```





Note

第 4 步，为列标题行定义渐变背景，同时增加高亮内阴影效果，为标题文本增加淡淡阴影色，代码如下。

```
.table thead th {
    text-shadow: 0 1px 1px rgba(0,0,0,.1);
    border-bottom: 1px solid #ccc;
    background-color: #eee;
    background-image: linear-gradient(to top, #f5f5f5, #eee);
}
```

第 5 步，设计数据隔行换色效果，代码如下。

```
.table tbody tr:nth-child(even) {
    background: #f5f5f5;
    box-shadow: 0 1px 0 rgba(255,255,255,.8) inset;
}
```

第 6 步，设计表格圆角效果，代码如下。

```
/* 左上角圆角 */
.table   thead th:first-child { border-radius: 6px 0 0 0;}
/* 右上角圆角 */
.table   thead   th:last-child {border-radius: 0 6px 0 0;}
/* 左下角圆角 */
.table tfoot td:first-child, .table tfoot th:first-child{ border-radius: 0 0 0 6px;}
/* 右下角圆角 */
.table tfoot td:last-child,.table tfoot th:last-child {border-radius: 0 0 6px 0;}
```



视频讲解

7.3.8 设计日历表

日历表在网页中经常看到，它适合使用表格结构进行设计。本例设计的是一个相对比较简单的日历表，其中有当天日期状态和文字说明，双休日以红色文字浅灰色背景显示，同时将周日到周一的标题加粗显示。

【操作步骤】

第 1 步，启动 Dreamweaver，新建网页，保存为 index.html，在<body>标签内输入以下代码。

```
<table>
    <caption>2017 年 7 月 1 日</caption>
    <thead>
        <tr>
            <th>日</th>
```





```
                <th>一</th>
                <th>二</th>
                <th>三</th>
                <th>四</th>
                <th>五</th>
                <th>六</th>
            </tr>
        </thead>
        <tbody>
<tr><td>29</td><td>30</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr>
<tr><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td></tr>
<tr><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td></tr>
<tr><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td></tr>
<tr><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td><td>1</td><td>2</td></tr>
<tr><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr>
        </tbody>
    </table>
```

日历表以表格结构形式表示，不仅在结构上表达了日历是一种数据型的结构，而且能更显著地在页面无 CSS 样式情况下表现日历表应该具有的结构，如图 7.21 所示。

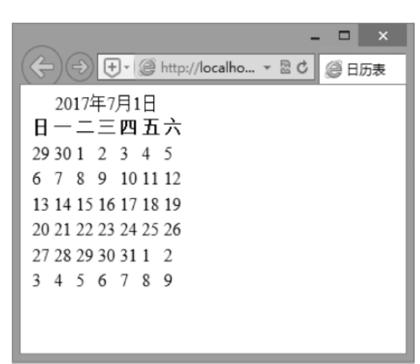


图 7.21　无 CSS 样式的日历表

第 2 步，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后输入下面样式，设计表格框样式。

```
table {/* 定义表格文字样式 */
 border-collapse:collapse; /* 合并单元格之间的边框 */
 border:1px solid #DCDCDC;
 font:normal 12px/1.5em Arial, Verdana, Lucida, Helvetica, sans-serif;
}
```





Note

合并表格单元格之间的边框，设计表格内对象的继承样式。例如，单元格之间的边框合并，文字样式。考虑日历表中显示的内容以数字居多，因此文字主要采用了英文字体。

第 3 步，设计表格标题样式。设置表头的高度属性和文字颜色，代码如下。

```
caption { /* 定义表头的样式，文字居中等 */
 text-align:center;
     line-height:46px;
     font-size:20px;
     color: blue;
}
```

第 4 步，设计单元格基本样式，代码如下。

```
td, th {/* 将单元格内容和单元格标题的共同点归为一组样式定义 */
 width: 40px;
     height: 40px;
     text-align: center;
     border: 1px solid #DCDCDC;
}
th {/* 针对单元格标题定义样式，使其与单元格内容产生区别 */
 color: #000000;
     background-color: #EEEEEE;
}
```

单元格内容 td 标签和单元格标题 th 标签所需要的样式只有背景颜色和文字颜色的不同，因此可以将这两个元素归为一个组定义样式，然后单独针对单元格标题定义背景颜色和文字颜色。这样的处理方式不仅减少了 CSS 样式的代码，也能使 CSS 样式代码更加直观，对于后期维护也会带来不少的帮助。

第 5 步，单元格<td>标签中所显示的时间是当前系统所显示的时间，添加一个名为 current 的 class 类名，并将其 CSS 样式定义成与其他单元格内容不同，突出显示当前日期。而且.current 类还有一个作用，就是为程序开发人员提供一个接口，方便他们在程序开发的过程中调用这个类名，便于判断系统当前日期后为页面实现效果，代码如下。

```
td.current {/* 定义当前日期的单元格内容样式 */
 font-weight:bold;
 color:#FFFFFF;
 background-color: blue;
}
```

第 6 步，设计.current 类之后，把该类绑定到表格当日单元格中，如<td class="current">1</td>。

第 7 步，日历表中为了能更好地体现某个月份的上一个月份的月尾几天和下一个月份的月头几天在当前月份中的位置，可以在页面中添加该内容，并通过 CSS 样式将其视觉效果弱化，代码如下。

```
td.last_month, td.next_month {color:#DFDFDF;} /* 定义上个月和下个月日期在当前月中的文字颜色 */
```



第 8 步，设计.last_month 和.next_month 类之后，把这两个类绑定到表格非当月单元格中，代码如下。

```
<tr>
    <td class="last_month">29</td>
    <td class="last_month">30</td>
    <td class="current">1</td>
    <td>2</td>
    <td>3</td>
    <td>4</td>
    <td>5</td>
</tr>
……
<tr>
    <td class="next_month">3</td>
    <td class="next_month">4</td>
    <td class="next_month">5</td>
    <td class="next_month">6</td>
    <td class="next_month">7</td>
    <td class="next_month">8</td>
    <td class="next_month">9</td>
</tr>
```

第 9 步，设计表格列组样式。在表格框<table>内部前面添加如下代码。

```
<table>
    <caption> 2017 年 7 月 1 日</caption>
    <colgroup span="7">
    <col span="1" class="day_off">
    <col span="5">
    <col span="1" class="day_off">
    </colgroup>
    <thead>
……
```

第 10 步，使用<colgroup>标签为表格的前后两列（即双休日）的日期定义一种样式，以区别于其他单元格内容中的日期，代码如下。

```
tr>td, tr>td+td+td+td+td+td+td {
    color:#B3222B;
    background-color:#F8F8F8;
} /* 定义第一列以及最后一列的单元格内容（即双休日）的样式 */
```



```
tr>td+td {
	color:#333333;
	background-color:#FFFFFF;
} /* 定义中间五列单元格内容的样式 */
col.day_off {
	color:#B3222B;
	background-color:#F8F8F8;
} /* 针对 IE 浏览器定义双休日的单元格样式 */
```

其中，tr>td 子选择符是将所有的单元格内容<td>标签设置文字颜色和背景颜色；tr>td+td+td+td+td+td+td 是子选择符与相邻选择符的结合，定义最后一列单元格内容<td>标签的文字颜色和背景颜色；再次定义 tr>td+td 是将除了第一列以外的所有单元格内容<td>标签定义样式，但因为 CSS 优先级的关系，无法覆盖最后一列单元格<td>标签的样式。最终形成的是前后两列的样式与中间五列的样式不同。

col.day_off 是针对 IE 浏览器而定义样式，主要是第一列与最后一列的文字颜色和背景颜色。该选择符的定义方式需要 XHTML 结构的支持，读者可以查看 XHTML 结构中<col>标签选择控制列的方式。

第 11 步，设计完毕，保存页面，在浏览器中预览，显示效果如图 7.22 所示。

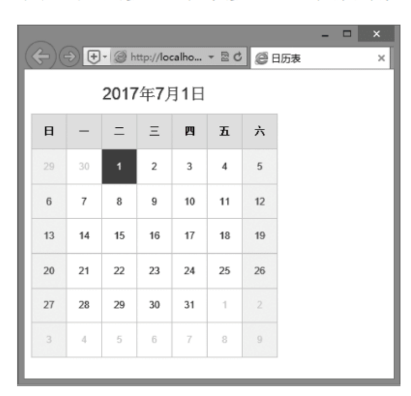


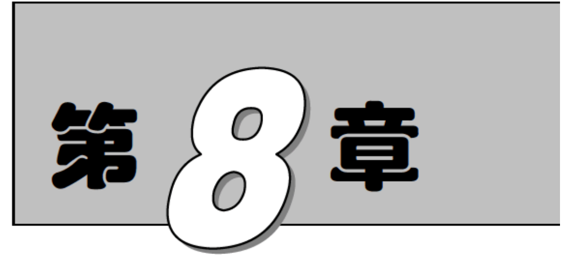
图 7.22　日历表页面设计效果

7.4 在线练习

下面通过大量的线上示例，帮助初学者练习使用 HTML5 设计表格结构和样式。感兴趣的读者可以扫码练习。







使用 CSS 美化表单

表单是网页交互的基础，网站一般都借助表单实现用户与服务器之间的信息交流，如注册表、登录表、调查表和留言表等。HTML5 新增了很多表单控件，完善了部分表单控件的功能，新特性提供了更好的用户体验和输入控制。

【学习要点】

- 正确使用各种表单控件。
- 掌握表单属性的设置。
- 设计易用性表单页面。





Note



视频讲解

8.1 HTML5 表单基础

与表格一样，表单也包含多个标签，它由很多控件构成，如文本框、文本区域、单选按钮、复选框、下拉菜单和按钮等。一般情况下，表单结构可分为以下 3 个部分。

☑ 表单框：使用<form>标签定义，主要功能是定义提交表单的处理方法、URL 和字符编码等。

☑ 表单对象：包括文本框、密码框、隐藏域、多行文本框、复选框、单选按钮、下拉选择框、文件上传框、提交按钮、复位按钮和一般按钮等。

☑ 辅助对象：包括提示性标签<label>、表单对象分组标签<fieldset>，用于表单结构的辅助设计。

【示例】 新建网页，保存为 test.html，在<body>内使用<form>标签包含两个<input>标签和一个提交按钮，并借助<p>标签把按钮和文本框分行显示，代码如下。

```
<form action="#" method="get" id="form1" name="form1">
    <p>用户名：<input name="" type="text" /></p>
    <p>密码：<input name="" type="text" /></p>
    <p><input type="submit" value="提交"/></p>
</form>
```

在 IE 浏览器中预览，演示效果如图 8.1 所示。



图 8.1　表单的基本效果

HTML5 定义了多个表单标签，简单说明如表 8.1 所示。

表 8.1　HTML 表单标签及其说明

标　签	说　明
<form>	定义供用户输入的 HTML 表单
<input>	定义输入控件
<textarea>	定义多行的文本输入控件
<button>	定义按钮
<select>	定义选择列表（下拉列表）
<optgroup>	定义选择列表中相关选项的组合
<option>	定义选择列表中的选项





续表

标　签	说　明
<label>	定义 input 元素的标注
<fieldset>	定义围绕表单中元素的边框
<legend>	定义 fieldset 元素的标题
<isindex>	定义与文档相关的可搜索索引，不赞成使用
<datalist>	HTML5 新增标签，定义下拉列表
<keygen>	HTML5 新增标签，定义生成密钥
<output>	HTML5 新增标签，定义输出的一些类型

<input>标签是通用输入型表单对象，使用它可以定义多种类型的表单对象，另外 HTML5 又扩展了很多输入型表单对象，简单说明如表 8.2 所示。

表 8.2　<input>标签可定义的输入型表单对象及其说明

表 单 对 象	说　明
<input type="text">	单行文本输入框
<input type="password">	密码输入框（输入的文字用点号表示）
<input type="checkbox">	复选框
<input type="radio">	单选按钮
<input type="file">	文件域
<input type="submit">	将表单（form）里的信息提交给表单属性 action 所指向的文件
<input type="reset">	将表单（form）里的信息清空，重新填写
<input type="color">	HTML5 新增对象，颜色选择器
<input type="date">	HTML5 新增对象，日期选择器
<input type="time">	HTML5 新增对象，时间选择器
<input type="datetime">	HTML5 新增对象，UTC 日期时间选择器
<input type="datetime-local">	HTML5 新增对象，本地日期时间选择器
<input type="week">	HTML5 新增对象，选择第几周的文本框
<input type="month">	HTML5 新增对象，月份选择器
<input type="email">	HTML5 新增对象，Email 输入框
<input type="tel">	HTML5 新增对象，电话号码输入框
<input type="url">	HTML5 新增对象，URL 输入框
<input type="number">	HTML5 新增对象，只能输入数字的文本框
<input type="range">	HTML5 新增对象，拖动条或滑块
<input type="search">	HTML5 新增对象，搜索文本框，与 type="text";定义的文本框没有太大区别




Note

8.2 案 例 实 战

CSS 没有为表单定义专用属性，不过可以使用 CSS 其他属性来设计表单样式，如字体、背景、颜色、边框、边距等。注意，由于部分表单对象是相对复杂的控件，如下拉菜单、文件域、复选框、单选按钮等，使用 CSS 可能无法完美控制其外观，必要时需要 JavaScript 脚本辅助实现。


视 频 讲 解

8.2.1 设计登录表单

本例将设计一款个性的登录表单页面，效果如图 8.2 所示。



图 8.2　设计网站登录页面效果

【操作步骤】

第 1 步，新建 HTML5 文档，保存为 index.html。打开网页文档，设计如下表单结构。

```
<form id="login-form" action="#" method="post">
    <fieldset>
        <legend>登录</legend>
        <label for="login">Email</label>
        <input type="text" id="login" name="login"/>
        <div class="clear"></div>
        <label for="password">密码</label>
        <input type="password" id="password" name="password"/>
        <div class="clear"></div>
        <label for="remember_me" style="padding: 0;">记住状态?</label>
        <input type="checkbox" id="remember_me" name="remember_me"/>
        <div class="clear"></div><br />
        <input type="submit" class="button" name="commit" value="登  录"/>
    </fieldset>
</form>
<p align="center"><strong>&copy; www.xxxxxx.cn</strong></p>
```





第 2 步，为 form 元素定义 id 属性，以便对整个表单进行控制，同时方便设计 ID 样式。

第 3 步，新建 CSS 样式表文件，命名为 style.css，保存到 images 文件夹中，然后在页面头部区域导入该样式表，代码如下。

```
<link rel="stylesheet" type="text/css" href="images/style.css" />
```

第 4 步，通过通配选择器清除页面中所有标签的内外边距，代码如下。

```
* { margin: 0; padding: 0; }
```

第 5 步，在 body 元素中定义网页字体效果，如类型、大小和颜色，设计网页背景图像，并让背景图像偏上居中显示，禁止平铺，同时设置背景图像无法覆盖的区域颜色显示为浅灰色（#c4c4c4），代码如下。

```
body { font-family: Georgia, serif; background: url(login-page-bg.jpg) center -50px no-repeat #c4c4c4; color: #3a3a3a; }
```

第 6 步，定义清除样式类，以便控制页面中每个表单域的换行显示，代码如下。

```
.clear { clear: both; }
```

第 7 步，设计表单对象样式。其中通过属性选择器控制复选框的样式，代码如下。

```
form { width: 406px; margin: 120px auto 0; }
legend { display: none; }
fieldset { border: 0; }
label { width: 115px; text-align: right; float: left; margin: 0 10px 0 0; padding: 9px 0 0 0; font-size: 16px; }
input { width: 220px; display: block; padding: 4px; margin: 0 0 10px 0; font-size: 18px; color: #3a3a3a; font-family: Georgia, serif; }
input[type=checkbox] { width: 20px; margin: 0; display: inline-block; }
```

第 8 步，设计按钮在鼠标指针经过和未经过时的状态样式，代码如下。

```
.button { background: url(button-bg.png) repeat-x top center; border: 1px solid #999; -moz-border-radius: 5px; padding: 5px; color: black; font-weight: bold; -webkit-border-radius: 5px; font-size: 13px; width: 70px; }
.button:hover { background: white; color: black; }
```

8.2.2 设计信息登记表



在设计表单时，正确选用各种表单控件很重要，这是结构标准化和语义化的要求，也是用户体验的需要。例如：

☑ 不确定答案可以建议用户输入，而不是让用户选择，如姓名、地址、电话等常用信息，使用输入的方式收集会比使用选择的方式收集更加自然且简单。

☑ 对于容易记错的答案不妨让用户选择，此时就不适合让用户使用输入框来输入，如国家、




Note

年、月、日、星座等，可以使用单选按钮组、复选框、列表框、下拉菜单等形式。

☑ 当设计选择项目时，如果希望用户浏览所有选项，则应该使用单选按钮组或复选框组，而不应该使用下拉菜单。下拉菜单会隐藏部分选项，对于用户来说，可能不会耐心地逐个浏览每个菜单项。

☑ 当选项很少时，不妨考虑使用单选按钮组或复选框组，而选项较多时，使用单选按钮组或复选框会占用很大的页面，此时不妨考虑使用下拉菜单。

☑ 多项选择可以有两种设计方法：使用复选框或者使用列表框。使用复选框比使用列表框更直观，有些用户不清楚列表框的作用和操作方法，这时就需要加上说明性文字，显然这样做没有使用复选框简单。

☑ 为控件设置默认值，建议采用一些提示性说明文字或常用值，能够提醒用户输入，这是一个很人性化的设计。

☑ 对于单选按钮组、复选框或下拉菜单，设计控件的 value 属性值或显示值应从用户的角度考虑，努力使用户浏览选项的时候更方便、简单，避免出现歧义或误解的值。

☑ 对于单选、复选的选项，应减少选项的数量，同时也可以使用短语来作为选项。

☑ 对于选项的排列顺序，最好遵循合理的逻辑顺序，如按首字母排列、按声母排列等，并根据普遍情况确定默认值。

☑ 设计表单时，应该避免使用多种表单控件，使用多种表单控件能够使页面看起来更好看，但实际上不利于用户的操作。

下面介绍如何设计一个信息登记表。

【操作步骤】

第 1 步，启动 Dreamweaver，新建 HTML5 文档，保存为 index.html。

第 2 步，在页面中构建 HTML 导航框架结构。切换到代码视图，在<body>标签内手动输入下面代码，定义表单框架结构。

```
<form action="#" class="form1">
    <p><em>*</em>号所在项为必填项</p>                <!-- 提示段落 -->
    <fieldset class="fld1">                          <!-- 字段集 1 -->
    <legend>个人信息</legend>                        <!-- 字段集 1 标题 -->
    <ol>                    <!-- 字段集 1 内嵌列表 -->
        <li>
            <label for="name">姓名<em>*</em></label>  <!-- 说明标签，以下类同 -->
            <input id="name">
        </li>
        <li>
            <label for="address">地址<em>*</em></label>
            <input id="address">
        </li>
        <li>
```





```
            <label for="dob">出生<span class="sr">日</span><em>*</em></label>
            <select id="dob">
                <option value="1">1</option>
                <option value="2">2</option>
            </select>
            <label for="dob-m" class="sr">月<em>*</em></label>
            <select id="dob-m">
                <option value="1">Jan</option>
                <option value="2">Feb</option>
            </select>
            <label for="dob-y" class="sr">年<em>*</em></label>
            <select id="dob-y">
                <option value="1979">1979</option>
                <option value="1980">1980</option>
            </select>
        </li>
        <li>
            <label for="sex">性别<em>*</em></label>
            <select id="sex">
                <option value="female">女</option>
                <option value="male">男</option>
            </select>
        </li>
    </ol>
    </fieldset>
    <fieldset class="fld2">                              <!-- 字段集 2 -->
    <legend>其他信息</legend>                            <!-- 字段集 2 标题 -->
    <ol>                                                 <!-- 字段集 1 内嵌列表 -->
        <li>
            <fieldset>                                   <!-- 列表内嵌字段集合 -->
            <legend>你喜欢这个表单吗? <em>*</em></legend><!--子字段集合标题 -->
            <label><input name="invoice-address" type="radio">喜欢</label>
            <label><input name="invoice-address" type="radio">不喜欢</label>
            </fieldset>
        </li>
        <li>
            <fieldset>
            <legend>你喜欢什么运动?</legend>
```

```
                <label for="football"><input id="football" type="checkbox">足球</label>
                <label for="golf"><input id="basketball" type="checkbox">篮球</label>
                <label for="rugby"><input id="ping" type="checkbox">乒乓球</label>
                </fieldset>
            </li>
            <li>
                <fieldset>
                <legend>请写下你的建议？<em>*</em></legend>
                <label for="comments"><textarea id="comments" rows="7" cols="25"></textarea></label>
                </fieldset>
            </li>
        </ol>
        </fieldset>
        <input value="提交个人信息" class="submit" type="submit">
    </form>
```

第 3 步，在<head>标签内输入<style type="text/css">，定义一个内部样式表，然后在<style>标签内手动输入下面样式代码。

```
body {/*定义页面基本属性*/
    font: normal 12px "宋体", Helvetica, Verdana, Arial;
    }
p { /*定义段落属性*/
    margin: 10px 0;
    text-align:right;
    }
ul, ol, dl, li, dt, dd {/*定义列表相关元素属性*/
    list-style-type:none;                /* 清除样式 */
    margin:0 0 0 1em;                    /* 清除边界，并定义左边界为 1 个字宽 */
    padding:0;                           /* 清除补白 */
    }
form {/*定义表单域基本属性*/
    padding:2em;                         /* 定义补白空隙 */
    border:solid 1px #E7F8C4;            /* 定义表单域边界 */
    text-align:center;                   /* 居中对齐，实现按钮居中显示 */
    }
fieldset {/*定义字段集基本属性*/
    text-align:left;                     /* 左对齐 */
    }
```





```
legend {/*定义字段集标题属性*/
    padding: 0;                    /* 清除补白 */
    margin:0;                      /* 清除边界 */
    color: #000;                   /* 标题颜色 */
    font-weight:bold;              /* 标题加粗显示 */
    }
li legend {/*定义列表内嵌字段集标题属性*/
    font-weight:normal;            /*清除加粗显示 */
    }
input, textarea, select {/*定义表单控件基本属性*/
    margin: 0;                     /* 定义边界为 0 */
    padding: 0;                    /* 定义补白为 0 */
    }
.sr {/*定义 label 内补充信息属性*/
    position: absolute;            /* 绝对定位 */
    left: -9999em;                 /* 隐藏显示，只对机器搜索使用 */
    }
form.form1 {/*定义表单属性*/
    width: 370px;                  /* 定义表单宽 */
    font-size: 1.1em;              /* 定义表单字体大小 */
    color: #333;                   /* 定义表单字体颜色 */
    background: #fff url(fieldset.gif) left bottom repeat-x; /* 定义表单背景图像 */
    }
form.form1 fieldset {/*定义字段集边框属性*/
    border: none;                  /* 清除边框 */
    border-top: 1px solid #C9DCA6; /* 显示顶部边框 */
    }
form.form1 .fld1 li {/*定义字段集 1 内的列表项的补白*/
    padding: 4px;}
form.form1 .fld2 li {/*定义字段集 2 内的列表项的补白*/
    padding: 2px;}
li   fieldset label {/*定义内嵌字段集的列表项左补白距离*/
    padding:0 0 0 2em;}
```

第 4 步，保存文档，按 F12 功能键，在浏览器中预览，效果如图 8.3 所示。





Note



图 8.3　定义表单结构



视频讲解

8.2.3　设计易用表单

表单设计应考虑用户的易用性，表单布局（如控件摆放位置、对齐方式、标签信息与周围元素）的设计会或多或少影响用户的操作。为此，在设计表单布局时，不妨从下面几个角度思考，来提高自己的设计水平。

1. 排列方式

根据习惯，表单控件一般使用垂直排列方式进行分布，这样能够加快视觉的移动和操作。水平排列容易使视觉移动起来很累，即使多列有规律的布局也是不可取的，人眼左右晃动操作很容易出错，如图 8.4 所示（index1.html）。

2. 控件分组

给控件分组也是表单布局中一个重要的技巧，特别是表单控件很多时，分组就显得很有必要，实际上分组是帮助用户进行逻辑梳理，避免混乱。

例如，图 8.5 所示（index2.html）表单域由于没有实现分组，看起来很容易混乱，这样就会影响操作速度，每填写一项信息都需要短暂的停留，并进行思考。如果将其分为 3 组：个人信息、地址和联系信息，就会使用户在填写表单时思路很清晰。

3. 缩进分布

当分组标题、控件和提示信息都并列排在一起时，很容易出现主次不分的情况，用户需要分辨哪些是操作的行，哪些是说明性文字，这样会影响操作速度。对此，可以采用缩进的方式，实现多层次叠进，帮助读者快速进行阅读，如图 8.6 所示（index3.html）。





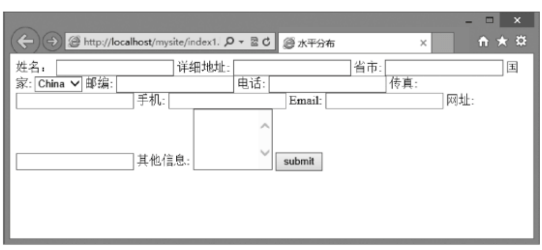




图 8.4　排列方式



图 8.5　控件分组

4. 标签突出

一般标签与控件水平并列分布是最佳的分布方式。部分用户喜欢使用垂直分布方式，即标签在上一行，控件在下一行，这种方式对于内容较少的表单域来说可行，但如果是一个大型表单，这种方式会消耗用户的视力，降低操作速度。

在表单布局中，推荐使用加粗的标签，这可以增加它们的视觉比重，提高其显著性。否则，从用户的角度分析，标签与输入框的文字有时会一样，可能会产生混淆的现象，如图 8.7 所示（index4.html）。



图 8.6　缩进分布



图 8.7　标签突出





5. 标签对齐

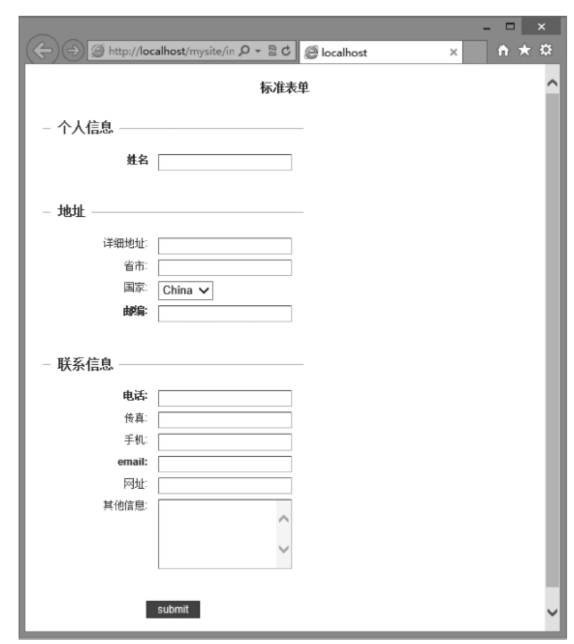
关于标签是左对齐还是右对齐的问题，一般来说，一致的左对齐可以减少眼睛移动和处理的时间。左对齐的标签还容易通览表单信息，用户只需要看左侧标签即可，而不会被控件打断思路。但这样也容易使标签与其对应的控件之间的距离被更长的标签拉大，从而影响操作表单的时间。用户必须左右来回移动视线找到两个对应的标签和控件，如图 8.8 所示（index5.html）。

而标签右对齐布局就会避免这个问题，使得标签和控件之间均匀分布并保持更紧密的联系，而这样分布的缺点是标签左边参差不齐的空白会影响用户快速检索表单填写内容。对此，可以根据实际情况有选择地使用标签左右对齐方式。

6. 背景和辅助线

上面所介绍的是一些基本的布局方法，实际上改善表单布局的方法还有很多，尝试为表单控件适当添加背景色和分割线，通过背景色和辅助线的视觉区分，也能加快用户操作速度，这对于划分操作区信息是很有效的。

背景色和线条对于区分表单的主要操作按钮尤其有效。但在使用这些辅助元素时，要避免它们影响用户的操作，因为色彩过浓的线条和背景色都能够分散用户的视线，过多的分格线会给用户阅读带来障碍，如图 8.9 所示（index6.html）。



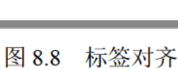


图 8.8　标签对齐

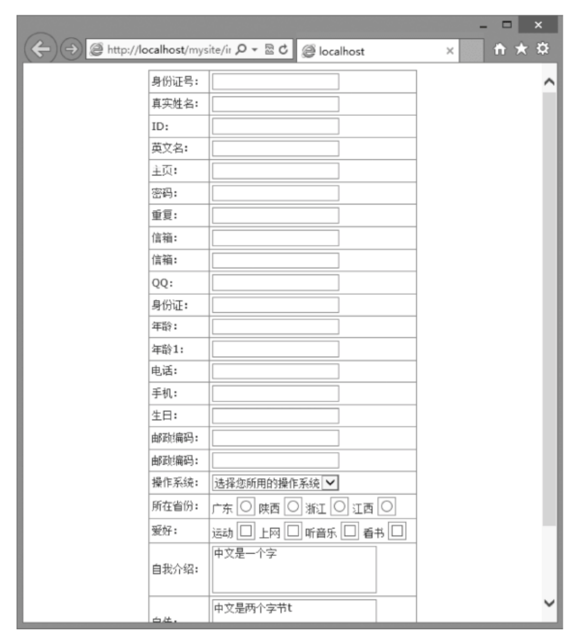


图 8.9　背景和辅助线

7. 动态效果

当用户选中或操作某个表单控件时，当前表单对象会显示为另一种样式，以区别其他控件。这个技巧对于用户的操作具有提示作用，避免出现用户有时不知当前操作的是哪个表单控件的情况，如图 8.10 所示（index7.html）。





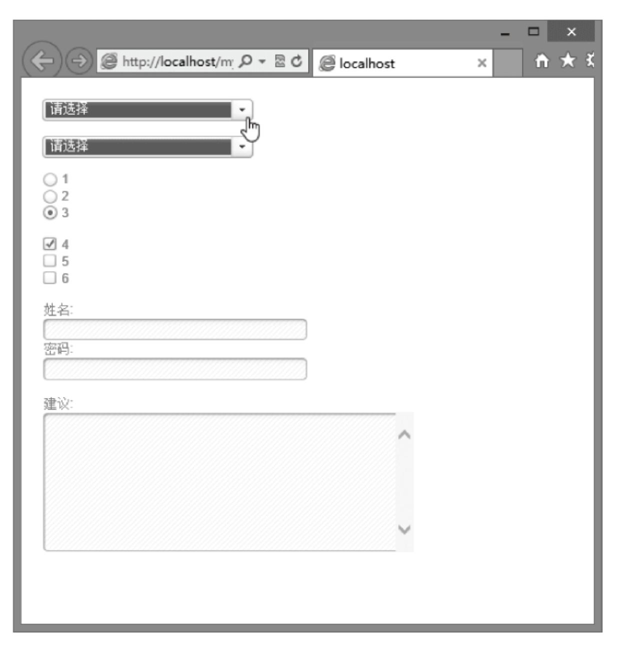


图 8.10　动态效果

当表单控件很多时，通过添加类样式，就可以让表单更具提示性，也使用户有更好的体验。为某个控件定义伪类，如:hover、:focus 和:focus:hover 属性样式，让输入框被鼠标激活时更加突出，利于用户集中精神填写。当然这对于老版本的 IE 浏览器没有作用，此时用户需要使用 JavaScript 脚本来控制。

8.2.4　设计注册表单

设计表单时，一般用户喜欢设计表单对象的边框，以便实现表单与页面整体效果的融合。CSS 盒模型适用任何表单对象，所以可以使用任何盒模型属性来定义表单对象。注意，除了 form 元素是块状元素之外，其他元素都以内联元素显示。

【操作步骤】

第 1 步，启动 Dreamweaver，新建 HTML5 文档，保存为 index.html。

第 2 步，在页面中构建 HTML 导航框架结构。切换到代码视图，在<body>标签内手动输入下面代码，定义表单框架结构。

```
<div id="box"><form  id=form1 action=#public method=post enctype=multipart/form-data>
    <h2>个人信息注册表单</h2>
    <ul>
        <li class="label">姓名
        <li> <input id=field1 size=20 name=field1>
        <li class="label">职业
        <li> <input name=field2 id=field2 size="25">
```




Note

```
            <li class="label">详细地址
            <li><input name=field3 id=field3 size="50">
            <li class="label">邮编
            <li><input name=field4 id=field4 size="12" maxlength="12">
            <li class="label">省市
            <li><input id=field5 name=field5>
            <li class="label">国家
            <li> <select id=field6 name=field6>
                    <option value=china>China</option>
                    <option value=armenia>Armenia</option>
                    <option value=australia>Australia</option>
                    <option value=italy>Italy</option>
                    <option value=japan>Japan</option>
                </select>
            <li class="label">Email
            <li><input id=field7 maxlength=255 name=field11>
            <li class="label">电话
            <li><input maxlength=3 size=6 name=field8>-
                <input maxlength=8 size=16 name=field8-1>
            <li class="label"><input id=saveform type=submit value=提交></li>
        </ul>
</form> </div>
```

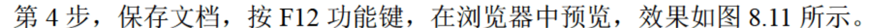
第 3 步，在<head>标签内输入<style type="text/css">，定义一个内部样式表，然后在<style>标签内手动输入下面样式代码。

```
body {
    margin: 0; padding:0;
    font-family: "lucida grande", tahoma, arial, verdana, sans-serif;
    }
#box{
    background:url(images/bg1.jpg);
    width:1015px; height:770px;
    }
#form1 {
    width:450px;
    text-align:left; font-size:12px;
    padding:12px 32px; margin:0 auto;
    }
```





```
#form1 h2 {
      border-bottom:dotted 1px #E37EA6;
      text-align:center;
      font-weight:normal;                         /* 清除标题加粗默认样式 */
      }
ul {/* 清除列表样式 */
      padding:0; margin:0;
      list-style-type:none;
      }
input {border:groove #ccc 1px; }                  /* 定义 3D 凹槽立体效果 */
.field6 { color:#666; width:32px;}
.label {
      font-size:13px; font-weight:bold;
      margin-top:0.7em;
      }
```



第 4 步，保存文档，按 F12 功能键，在浏览器中预览，效果如图 8.11 所示。



图 8.11　设置表单样式

8.2.5　设计联系表单



本例设计一个联系表单，使用 CSS 背景图像来设计表单样式，使其更具艺术性。

【操作步骤】

第 1 步，启动 Dreamweaver，新建 HTML5 文档，保存为 index.html。

第 2 步，在页面中构建 HTML 导航框架结构。切换到代码视图，在<body>标签内手动输入下面



代码，定义表单框架结构。



```
<form id="fieldset" action="default.asp" method="post">
    <h2>联系表单</h2>
    <label for=name>姓名</label>
    <input class="textfield" id="name" name="name"><br>
    <label for=email>Email</label>
    <input class="textfield" id="email" name="email"><br>
    <label for=website>网址</label>
    <input class="textfield" id="website" value="http://" name="website"><br>
    <label for=comment>反馈</label>
    <textarea class="textarea" id="comment" name="comment" rows="15" cols="30"></textarea><br>
    <label for=submit> </label>
    <input class="submit" id="submit" type="submit" value="提交" name="submit">
</form>
```

第 3 步，在<head>标签内输入<style type="text/css">，定义一个内部样式表，然后在<style>标签内手动输入下面样式代码。

```
body {/*定义页面属性 */
    font-size: 12px;                          /* 定义字体大小*/
    margin: 50px;                             /* 定义边界，避免顶部跑到页面外边 */
    color: #666;                              /* 定义颜色 */
    font-family: 宋体, verdana, arial, helvetica, sans-serif;/* 定义字体 */
    }
#fieldset {/*定义表单属性 */
    border: #fff 0px solid;                   /* 清除边框 */
    width: 300px;                             /* 定义表单域宽度 */
    background-color: #ccc;                   /* 定义浅灰色背景 */
    }
#fieldset h2 {/*定义表单标题属性 */
    padding: 0.2em;                           /* 定义补白，增加边缘空隙 */
    margin:0;                                 /* 清除标题预定义边界 */
    position: relative;                       /* 相对定位 */
    top: -1em;                                /* 在现有流位置向上移动一个字体距离 */
    background: url(h2_bg.gif) no-repeat;     /* 定义背景图像，圆角显示 */
    width: 194px;                             /* 定义宽度，该宽度与背景图像宽相同 */
    font-size: 2em;                           /* 定义字体大小 */
    color: #fff;                              /* 定义字体颜色 */
```






```
        white-space: pre;                              /* 保留标题预定义格式，可以保留多行显示 */
        letter-spacing: -1px;                          /* 收缩字距 */
        text-align:center;                             /* 居中显示 */
        }
#fieldset label {/*定义表单标签属性 */
        padding: 0.2em;                                /* 增加边距空隙 */
        margin: 0.4em 0px 0px;                         /* 增加顶部边界，加大与上一个控件的间距 */
        float: left;                                   /* 向左浮动 */
        width: 70px;                                   /* 定义宽度 */
            text-align: right;                         /* 右对齐 */
          }
.br {/*隐藏换行标签，也不占据位置 */
        display: none;
        }
.textfield {/*定义输入表单控件 */
        border: #fff 0px solid;                        /* 清除边框 */
        padding: 3px 8px;                              /* 增加内容边距空隙 */
        margin: 3px;                                   /* 定义边界距离 */
        width: 187px;                                  /* 定义宽 */
        height: 20px;                                  /* 定义高 */
        background: url(textfield_bg.gif) no-repeat;   /* 定义输入表单控件背景图像 */
        color: #FF00FF;                                /* 定义表单显示字体颜色 */
        font: 1.1em verdana, arial, helvetica, sans-serif;  /* 定义字体属性 */
        }
textarea {/*定义文本域控件属性 */
        border: #fff 0px solid;                        /* 清除边框 */
        padding:4px 8px;                               /* 增加内容边距，避免内部文本顶到边框 */
        margin: 3px;                                   /* 定义边界距离 */
        height: 150px;                                 /* 定义高 */
        width: 190px;                                  /* 定义宽 */
        background: url(textarea_bg.gif) no-repeat;    /* 定义文本域表单控件背景图像 */
        color: #FF00FF;                                /* 定义表单显示字体颜色 */
        font: 1.1em verdana, arial, helvetica, sans-serif;  /* 定义字体属性 */
        }
.submit {/*定义按钮控件属性 */
        border: #fff 0px solid;                        /* 清除边框 */
        margin: 6px;                                   /* 定义边界距离 *
        width: 80px;                                   /* 定义宽 */
```




Note

```
height: 20px;                                  /* 定义高 */
background: url(submit.gif) no-repeat;         /* 定义按钮控件背景图像 */
text-transform: uppercase;                     /* 英文大写显示 */
font: 1.1em verdana, arial, helvetica, sans-serif;   /* 定义字体属性 */
color: #666;                                   /* 定义字体颜色 */
}
```

第 4 步，保存文档，按 F12 功能键，在浏览器中预览，效果如图 8.12 所示。



图 8.12　设置表单背景样式

关于背景图像的应用还是比较灵活的，用户可以充分发挥想象力，设计出更具创意的表单效果。例如，制作动态表单，先制作好动态的 GIF 图像，然后引入即可。


视 频 讲 解

8.2.6　设计高亮样式

本例借助 CSS 的 UI 伪类选择器设计在表单对象获取焦点和失去焦点时，分别为该表单对象应用不同的样式，从而实现当前表单高亮显示效果，如图 8.13 所示。



图 8.13　高亮提示表单效果





【操作步骤】

第 1 步，新建文档，在<body>标签中输入下面代码，构建一个表单结构，该结构是在上一节示例基础上重新定义的。

```
<form id="form1" name="form1" method="post" action="">
    <fieldset>
    <legend>高亮提示表单</legend>
    <label for="name" class="title">姓名：</label>
    <input size="40" name="name" class="input1">
    <label for="email" class="title">邮箱：</label>
    <input size="40" name="email" class="input1">
    <label for="url" class="title">网址：</label>
    <input size="40" name="url" class="input1">
    <label for="subject" class="title">主题：</label>
    <input size="40" name="subject" class="input1">
    <label for="message" class="title">内容：</label>
    <textarea name="message" cols="39" rows="5" class="input1" ></textarea>
    </fieldset>
</form>
```

第 2 步，在<head>标签内插入<style>标签，定义内部样式表。

第 3 步，定义 CSS 样式控制表单的显示，同时定义两个类，以供 JavaScript 属性事件调用，代码如下。

```
body {/* 统一网页字体大小 */
    font-size:12px;                     /* 12 像素字体大小 */
}
.title {/* 提示文本样式类 */
    width:100px;                        /* 固定宽度 */
    float:left;                         /* 向左浮动定位 */
    text-align:right;                   /* 文本右对齐 */
    font-weight:bold;                   /* 字体加粗 */
    margin:6px 0;                       /* 定义提示文本的外边距 */
}
input, textarea {/* 表单对象默认显示样式 */
    background-color: #EEEEEE;
    border-bottom: #FFFFFF 1px solid;
    border-left: #CCCCCC 1px solid;
    border-right: #FFFFFF 1px solid;
```




```
        border-top: #CCCCCC 1px solid;
    }
    input:focus, textarea:focus {/* 表单获取焦点时的样式 */
        background-color:#F0F8FF;
        border: 1px solid #999;
    }
```

Note



视 频 讲 解

8.2.7 设计图标表单

本例介绍如何把一个外部图标固定在文本框的左侧，使其既能点缀表单，又能够提示用户输入，演示效果如图 8.14 所示。

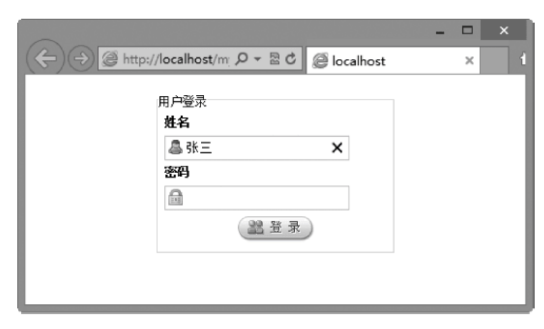


图 8.14　图标样式的表单效果

【操作步骤】

第 1 步，新建文档，在<body>标签中输入下面代码，构建一个表单结构。该结构依然保留上面示例的结构雏形，并适当进行增删。

```
<div id="login">
    <fieldset>
    <legend>用户登录</legend>
    <form action="" method="POST" class="form">
        <label for="name">姓名</label>
        <div><input name="name" type="text"  id="name" value="" /></div>
        <label for="password">密码</label>
        <div><input name="password" type="text" id="password" value="" /></div>
        <div class="button_div"><input type="image" src="images/login.gif" /></div>
    </form>
    </fieldset>
</div>
```

第 2 步，在<head>标签内插入<style>标签，定义内部样式表。

第 3 步，输入下面样式代码，使用 CSS 对这个表单进行布局。





```
* {/* 清除所有元素的边距 */
    margin:0;                        /* 清除内边距 */
    padding:0;                       /* 清除外边距 */
}
body {/* 定义网页基本属性 */
    text-align:center;               /* 网页居中显示 */
}
#login {/* 表单包含框样式 */
    margin:10px auto 10px;           /* 网页居中显示 */
    text-align:left;                 /* 文本左对齐 */
}
fieldset {/* 表单域样式 */
    width:230px;                     /* 固定表单的宽度 */
    margin:20px auto;                /* 定义表单的外边距 */
    font-size:12px;                  /* 统一表单的字体大小 */
}
```

第 4 步，要想让表单对象的提示文本与表单对象上下排列显示，需定义标签元素为块状元素，并定义它们的宽度为固定显示。详细代码如下。

```
label { /* 定义标签提示文本的样式 */
    width:200px;                 /* 固定宽度 */
    height:26px;                 /* 固定高度 */
    line-height:26px;            /* 固定行高 */
    text-indent:6px;             /* 文本首行缩进 6 像素 */
    display:block;               /* 块状显示 */
    font-weight:bold;            /* 加粗提示文本 */
}
```

第 5 步，定义表单对象的样式。先利用分组统一所有表单对象的样式，然后利用背景图像的方法单独为每个文本框左侧定义一个图标。为了避免文本框内的文本遮盖背景图像图标，需要定义左侧内边距以挤出一个空间给背景图像留用。详细代码如下。

```
#name, #password { /* 统一表单对象的样式 */
    border:1px solid #ccc;       /* 定义表单对象的边框样式 */
    width:160px;                 /* 固定宽度 */
    height:22px;                 /* 固定高度 */
    margin-left:6px;             /* 定义左侧外边距 */
    padding-left:20px;           /* 定义左侧内边距，挤出定义背景图像的空间 */
    line-height:20px;            /* 定义行高 */
```





Note

```
}
#name { /* 用户名文本框图标样式 */
    background:url(images/name.gif) no-repeat 2px 2px; /* 定义用户名文本框图标 */
}
#password { /* 用户密码文本框图标样式 */
    background:url(images/password.gif) no-repeat 2px 2px;    /* 定义密码文本框图标 */
}
.button_div { /* 按钮样式 */
    text-align:center;                          /* 按钮文本居中 */
    margin:6px auto;                            /* 按钮居中显示 */
}
```



视频讲解

8.2.8 设计反馈表

本例设计一个简单的反馈表，主要使用表单域<fieldset>标签、表单域标题<legend>标签、文件上传控件 input（type="file"）和文本域<textarea>标签。表单域<fieldset>标签主要是将表单分成多个小区域显示在网页中；表单域标题<legend>标签则是针对每个不同的表单域设置标题；文件上传控件 input（type="file"）结合后台开发程序或者 JavaScript 实现文件上传功能；文本域<textarea>标签是可以输入多行文本的元素控件。

【操作步骤】

第 1 步，新建文档，设计表单结构，代码如下，在浏览器中的显示效果如图 8.15 所示。

```
<div class="feedback">
 <h3>反馈表单</h3>
 <div class="content">
    <form method="post" action="">
        <fieldset class="base_info">
            <legend>用户信息</legend>
            <div class="frm_cont userName"><label for="userName">用户名：</label><input type="text"
value="" id="userName" /></div>
            <div class="frm_cont email"><label for="email">电子邮件：</label><input type="text"
value="@" id="email" /></div>
            <div class="frm_cont url"><label for="url">网址：</label><input type="text" value="http://"
id="url" /></div>
        </fieldset>
        <fieldset class="feedback_content">
            <legend>反馈内容</legend>
            <div class="frm_cont up_file">
                <label for="up_file">相关图片：</label><input type="file" id="up_file" />
```





```
                <p class="tips">本系统只支持上传.jpg、.gif、.png 图片。</p>
            </div>
            <div class="frm_cont msg">
                <label for="msg">内容：</label><textarea rows="4" cols="40" id="msg"></textarea>
                <p class="tips">请输入留言内容！</p>
            </div>
        </fieldset>
        <div class="btns"><button type="submit">提交</button><button type="reset">重置</button></div>
    </form>
 </div>
</div>
```

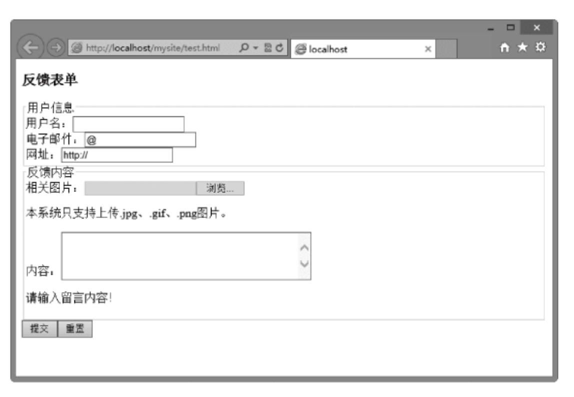


图 8.15　默认解析表单效果

第 2 步，使用 CSS 样式定义 HTML 标签的表现时，基本原则都是从外到内、从泛到细，更重要的是要善于利用 CSS 选择器，代码如下。

```
.feedback { /*  定义表单整体的宽度和边框样式等  */
    width:398px;
    padding:1px;
    border:1px solid #E8E8E8;
    background-color:#FFFFFF;
}
.feedback * { /*  定义表单内部的所有元素内补丁、外补丁以及文字的相关样式  */
    margin:0;
    padding:0;
    font:normal 12px/1.5em "宋体", Verdana,Lucida, Arial, Helvetica, sans-serif;
}
```

第 3 步，整体样式的定义主要包含反馈表单的整体宽度和内部所有子元素的整体定义。整体宽度、边框等样式的定义根据视觉效果而设定；定义内部所有子元素的样式是为了提高后期对子元素



样式定义的便利性，代码如下。



Note

```
.feedback h3 { /* 定义表单标题的高度、文字样式以及背景颜色等 */
    height:24px;
    line-height:24px;
    font-weight:bold;
    font-size:13px;
    text-indent:12px;
    color:#FFFFFF;
    background-color:#999999;
}
```

第 4 步，定义反馈表单标题的高度，并设置标题文本缩进和文字大小等样式，增强标题与内容之间的反差和整齐感，代码如下。

```
.feedback .content {/* 表单内容区域增加 10px 的左右内补丁，使其与表单外框产生间距 */
    padding:0 10px;
}
```

第 5 步，为了不让反馈表单内部信息与边框太紧密，将表单内容区域增加 10px 的左右内补丁，使其与表单整体有一定的间距，代码如下。

```
.feedback fieldset {/* 定义表单域边框样式以及与上下几个元素之间的间距 */
    padding-left:12px;              /* 因为前面已经将表单内部所有内补丁设置为 0，所以增加 12px 的
                                    左内补丁使表单域标题缩进 */
    margin-top:10px;
    border:0 none;                  /* 去除默认的表单域边框 */
    border-top:1px solid #999999;   /* 定义表单域上边框的样式 */
}
.feedback legend {
    padding:0 5px;                  /* 设置表单域的标题在表单域上边框中的间距 */
    color:#333333;                  /* 考虑 IE 浏览器解析表单域标题时文字颜色与 Firefox 浏览器不
                                    同，所以统一定义相同的颜色值 */
}
```

表单域在浏览器默认解析的情况下是有边框线的，不需要边框线就需要将其隐藏，需要部分边框线就要将不需要的部分隐藏。只需要一条上边框线，因此首先将所有边框消除（border:0 none;），然后再次定义上边框的样式。

第 6 步，在定义整体样式时，将所有内补丁（padding）定义为 0，导致表单域标题<legend>标签中的文字紧挨表单域边框。需要将其缩进就要增加左内补丁值。例如，padding-left:12px;即可将表单域标题缩进，使表单域标题占据在表单域上边框线之间，如图 8.16 所示为两者对比。



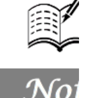




图 8.16　表单域缩进前后对表单域标题的影响对比

设置代码如下。

```
.feedback .frm_cont {
    margin-top:8px; /*  表单内容区域中不同表单之间的上下间距  */
}
```

第 7 步，将每个表单的内容增加上外补丁，增加每个表单元素之间的空间感，代码如下。

```
.feedback label { /*  定义 label 标签的宽度、右对齐，设置浮动，使其与输入框并列  */
    float:left;
    width:80px;
    height:22px;
    line-height:24px;
    text-align:right;
    color:#ABABAB;
    cursor:pointer;
}
```

第 8 步，<label>标签使用浮动后可以将旁边的元素（即文本输入框）“吸”到它的旁边，并设置了宽度和高度属性，再将文字右对齐。这样的排列效果在视觉效果上可以达到整齐的感觉，不会让浏览者感觉这个表单是杂乱无章的，代码如下。

```
.feedback .base_info input { /*  定义表单内容区域中所有输入框的宽度和高度等样式  */
    width:100px;
    height:17px;
    padding:3px 2px 0;
    border:1px solid #DEDEDE;
}
.feedback .email input { /*  针对 Email 地址输入框，改变其宽度属性值  */
    width:150px;
}
```



Note

```
.feedback .url input { /* 针对网址输入框，改变其宽度属性值 */
    width:240px;
}
/* 避免修改文件上传浏览框因输入框的高度和宽度修改导致浏览器之间的差别，使用 auto 默认值恢复浏览器默认解析 */
.feedback .up_file input {
    width:auto;
    height:auto;
}
```

.feedback .base_info input 选择器为反馈表单中类名为 base_info 的容器内所有的<input>标签设置了宽度、高度和边框样式，再针对不同功能的输入框设置宽度，不仅能加大显示输入数据的空间，还可以形成表单之间有序的错落感。

第 9 步，文件上传控件 input（type="file"）也是<input>标签，但不在类名为 base_info 的容器之内，所以最终显示的还是默认的浏览器解析效果，代码如下。

```
.feedback .tips { /* 将提示文本利用内补丁缩进，并设置红色，突出显示 */
    padding:5px 0 0 80px;
    color:#FF3260;
}
.feedback textarea {/* 定义文本域的宽高和内部文字的行高等样式 */
    width:240px;
    height:66px;
    padding-left:2px;
    line-height:22px;
    border:1px solid #DEDEDE;
}
.feedback .btns { /* 按钮区域增加上下内补丁，加大间距，并定义其内部的元素居中显示 */
    padding:5px 0;
    text-align:center;
}
.feedback .btns button { /* 定义按钮和按钮中文字的间距等样式 */
    height:22px;
    margin:0 5px;
    letter-spacing:3px; /* 调整文字间距 */
    padding-left:3px; /* 添加左内补丁使按钮左右间距相等 */
    cursor:pointer;
}
```

第 10 步，为文本域、提示信息和按钮等元素定义相关样式。文本域中使用 padding-left:2px;是需





要使文字与其边框产生间距；按钮中定义 letter-spacing:3px;可以让按钮中的文字之间有 3px 的间距，以文字的右边为基准。

第 11 步，将经过以上 CSS 修饰的 HTML 结构在浏览器中预览，页面效果如图 8.17 所示。



图 8.17　最终的反馈表单效果

8.2.9　设计搜索表单



搜索框一般包含关键词输入框、搜索类别、搜索提示和搜索按钮，当然简单的搜索框只有关键词输入框和搜索按钮两部分。本案例将介绍如何设计附带提示的搜索框样式，演示效果如图 8.18 所示。



图 8.18　设计搜索框

【操作步骤】

第 1 步，新建一个网页，保存为 index.html，在<body>标签内输入如下结构代码，构建表单结构。

```
<div class="search_box">
    <h3>搜索框</h3>
    <div class="content">
```




Note

```
            <form method="post" action="">
                <select>
                    <option value="1">网页</option>
                    <option value="2">图片</option>
                    <option value="3">新闻</option>
                    <option value="4">MP3</option>
                </select>
                <input type="text" value="css" /> <button type="submit">搜索</button>
                <div class="search_tips">
                    <h4>搜索提示</h4>
                    <ul>
                        <li><a href="#">css 视频</a><span>共有 589 个项目</span></li>
                        <li><a href="#">css 教程</a><span>共有 58393 个项目</span></li>
                        <li><a href="#">css+div</a><span>共有 158393 个项目</span></li>
                        <li><a href="#">css 网页设计</a><span>共有 58393 个项目</span></li>
                        <li><a href="#">css 样式</a><span>共有 158393 个项目</span></li>
                    </ul>
                </div>
            </form>
        </div>
    </div>
```

整个表单结构分为两个部分，将下拉选择、文本框和按钮归为一类，主要功能是搜索信息；搜索提示为当在文本框中输入文字时，将会出现相对应的搜索提示信息，该功能主要是由后台程序开发人员实现，前台设计师只需要将其以页面元素表现即可。

第 2 步，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后逐步输入 CSS 代码，设计表单样式。

第 3 步，通过分析最终效果可以看到，页面中并没有显示“站内搜索”和“搜索提示”这两个标题，且搜索按钮是以图片代替的，搜索提示出现在搜索输入框的底部，并且宽度与输入框相等。为此，在内部样式表中输入下面样式，对表单结构进行初始化设计。

```
.search_box { /* 设置输入框宽度，并设置为相对定位，成为其子级元素的定位参考 */
    position:relative;
    width:360px;
}
.search_box * { /* 设置输入框内补丁、边界为 0，列表修饰为无，并且设置字体样式等 */
    margin:0; padding:0;
    list-style:none;
    font:normal 12px/1.5em "宋体", Verdana,Lucida, Arial, Helvetica, sans-serif;
}
.search_box h3, .search_tips h4 {display:none; } /* 隐藏标题文字 */
```





第 4 步，设置搜索框整体的宽度属性值以及其所有子元素的内补丁、边界等相关属性。为了方便将搜索提示信息框通过定位的方式显示在搜索输入框的底部，需在.search_box 中定义 position 属性，让其成为子级元素定位的参照物。文档结构中的标题在页面中不需要显示，因此可以将其隐藏。在后期网站开发过程中如果需要显示，可以直接通过 CSS 样式修改，而不需要再次调整文档结构，代码如下。

```
.search_box select {/* 将下拉框设置浮动，并设置其宽度值 */
    float:left;
    width:60px;
}
.search_box input {/* 设置搜索输入框样式，浮动显示，添加左右两边间距（边界）*/
    float:left;
    width:196px; height:14px;
    padding:1px 2px; margin:0 5px;
    border:1px solid #619FCF;
}
.search_box button {/* 设置按钮浮动，以缩进方式隐藏按钮文字，添加背景图 */
    float:left;
    width:59px; height:18px;
    text-indent:-9999px;
    border:0 none;
    background:url(images/btn_search.gif) no-repeat 0 0;
    cursor:pointer;
}
```

第 5 步，搜索类别下拉框、搜索关键字输入框和搜索按钮这 3 个元素按照常理来理解原本就是可以并列显示的，但为了将这 3 个元素之间的默认空间缩短，可使用 float:left;来实现。再利用输入框 input 增加可控的边界 margin:0 5px; 调整三者之间的间距。

三者之间整体样式调整完毕后，再对其细节部分进行详细的调整修饰。美化输入框并且利用文字缩进属性隐藏按钮上的文字，使用图片代替。

第 6 步，下拉框<select>标签只是设置了宽度属性值，并未设置其高度属性值，其原因就是 IE 浏览器和 Firefox 浏览器对其高度属性值的解析完全不一样，因此采用默认的方式而不是再次利用 CSS 样式定义其相关属性。

第 7 步，按钮<button>标签在默认情况下不具备当光标悬停时显示手型，因此需要特殊定义，代码如下。

```
.search_tips { /* 将搜索提示框的宽度设置为与输入框相同，并绝对定位在输入框底部 */
    position:absolute;
    top:17px; left:65px;
    width:190px; padding:5px 5px 0;
    border:1px solid #619FCF;
}
```




第 8 步，搜索提示框使用绝对定位的方式显示在输入框的底部，其宽度属性值等于输入框的宽度属性值，可以提高视觉效果。不设置提示框的高度属性值是希望搜索框能随着内容的增加而自适应高度，代码如下。

```
.search_tips li {/* 设置搜索提示框内的列表高度和高度值，利用浮动避免 IE 浏览器中列表上下间距增多的bug*/
    float:left;
    width:100%;
    height:22px;
    line-height:22px;
}
```

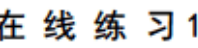
第 9 步，在 IE 早期版本浏览器中，列表<li>标签上下间距会增大显示，为了避免该问题的出现，将所有列表<li>标签添加浮动 float 属性。宽度属性值设置为 100%可以避免当列表<li>标签具有浮动属性时，宽度自适应的问题，代码如下。

```
.search_tips li a { /* 搜索提示中相关文字居左显示，并设置相关样式 */
    float:left;
    text-decoration:none;
    color:#333333;
}
.search_tips li a:hover { /* 搜索提示中相关文字在光标悬停时显示红色文字 */
    color:#FF0000;
}
.search_tips li span { /* 以灰色弱化搜索提示相关数据，并居右显示 */
    float:right;
    color:#CCCCCC;
}
```

第 10 步，将列表项<li>标签中的锚点<a>标签和<span>标签分别左右浮动，使它们靠两边显示在搜索提示框内，并相应地添加文字样式，做细节上的调整。

8.3 在线练习

1. 练习表单结构设计和基本行为的控制。
2. 练习使用 CSS3 设计各种网页表单特效。



在线练习1


在线练习2



第9章 使用 DIV+CSS 设计网页

在标准化设计中，网页内容主要通过 HTML 标签组织显示，然后使用 CSS 设计版式。本章将围绕 HTML 结构和 CSS 布局相关概念、思路和方法等基础知识进行讲解，并通过大量案例训练读者使用 CSS 进行网页布局。

【学习要点】

- 能够正确设计文档结构。
- 了解 CSS 盒模型。
- 设计边框样式。
- 设计边界和补白样式。
- 使用 CSS 浮动和定位技术排版页面内容。





Note



视频讲解

9.1 设计文档结构

搭建良好的网页结构是 CSS 布局的基础，如果 HTML 文档结构混乱，那么很多事情做起来都是很麻烦的。结构简洁、富有语义的结构会让后期排版更加轻松。

9.1.1 定义文档结构

一个完整的 HTML 文档必须包含 3 个部分：一个用<html>标签定义的文档容器，一个用<head>定义各项声明的文档头部和一个由<body>定义的文档主体部分。<head>作为各种声明信息的包含框，出现在文档的头部，并且要先于<body>出现。而<body>用来显示文档主体内容。

在 HTML 文档的头部区域，一般需要包括网页标题、基础信息和元信息等。HTML 的头部元素以<head>为开始标记，以</head>为结束标记。

<head>的作用范围是整篇文档，其中可以有<meta>元信息定义、文档样式表定义和脚本等信息，定义在 HTML 文档头部的内容往往不会在网页上直接显示。

网页的主体部分包括要在浏览器中显示处理的所有信息。在网页的主体标记中有很多属性设置，包括网页的背景设置、文字属性设置和链接设置等。

【示例】本示例是一个很简单的 HTML 文件，使用了尽量少的 HTML 标签。它演示了一个简单的文档应该包含的内容，以及 body 内容是如何在浏览器中显示的。

第 1 步，新建文本文件，输入下面的代码。

```
<html>
    <head>
        <meta charset="utf-8">
        <title>一个简单的文档包含内容</title>
    </head>
    <body>
        <h1>我的第一个网页文档</h1>
        <p>HTML 文档必须包含三个部分：</p>
        <ul>
            <li>html——网页包含框</li>
            <li>head——头部区域</li>
            <li>body——主体内容</li>
        </ul>
    </body>
</html>
```

第 2 步，保存文本文件，命名为 test，设置扩展名为.html。

第 3 步，使用浏览器打开这个文件，则可以看到如图 9.1 所示的预览效果。



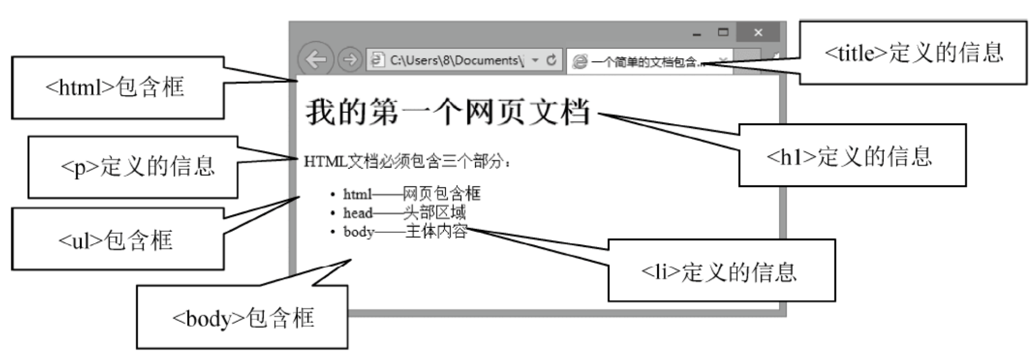


图 9.1　网页文档演示效果

> 注意：HTML 包含丰富的语义元素，正确选用它们可以避免代码冗余。在制作网页时不仅需要使用<div>标签来构建网页结构，还要使用下面几类标签完善网页结构。
>
> ☑ <h1>、<h2>、<h3>、<h4>、<h5>、<h6>：定义文档标题，1 表示一级标题，6 表示六级标题，常用标题包括一级、二级和三级。
> ☑ <p>：定义段落文本。
> ☑ 、<ol>、<li>等：定义信息列表、导航列表、榜单结构等。
> ☑ <table>、<tr>、<td>等：定义表格结构。
> ☑ <form>、<input>、<textarea>等：定义表单结构。
> ☑ <span>：定义行内包含框。

为了更好地选用标签，读者可以参考 w3school 网站的 http://www.w3school.com.cn/tags/index.asp 页面信息。其中，DTD 列描述标签在哪一种 DOCTYPE 文档类型是允许使用的：S=Strict，T=Transitional，F=Frameset。

9.1.2　使用 div 和 span



文档结构基本构成元素是 div，div 表示区块（division）的意思，它提供了将文档分割为有意义的区域的方法。通过将主要内容区域包围在 div 中并分配 id 或 class，就可以在文档中添加有意义的结构。

【示例 1】应该避免不必要的嵌套。例如，如果设计导航列表，就没有必要将<ul>再包裹一层<div>标签，代码如下。

```
<div id="nav">
    <ul>
        <li><a href="#">首页</a></li>
        <li><a href="#">关于</a></li>
        <li><a hzef="#">联系</a></li>
    </u1>
</div>
```

可以完全删除 div，直接在 ul 上设置 id，代码如下。





```
<ul id="nav">
    <li><a href="#">首页</a></li>
    <li><a href="#">关于</a></li>
    <li><a hzef="#">联系</a></li>
</ul>
```

过度使用 div 是结构不合理的一种表现，也容易造成结构复杂化。

与 div 不同，span 元素可以用来对行内元素进行分组。

【示例 2】下面的代码可将段落文本中的部分信息进行分隔显示，以便应用不同的类样式。

```
<h1>新闻标题</h1>
<p>新闻内容</p>
<p>......</p>
<p>发布于<span class="date">2016 年 12 月</span>，由<span class="author">张三</span>编辑</p>
```

对行内元素进行分组的情况比较少，所以使用 span 的频率没有 div 多。一般应用类样式时才会用到。

9.1.3 使用 id 和 class

HTML 是简单的文档标识语言，而不是界面语言。文档结构大部分使用<div>标签来完成，为了能够识别不同的结构，一般通过定义 id 或 class 给它们赋予额外的语义，给 CSS 样式提供有效的“钩子”。

【示例 1】构建一个简单的列表结构，并给它分配一个 id，自定义导航模块。

```
<ul id="nav">
    <li><a href="#">首页</a></li>
    <li><a href="#">关于</a></li>
    <li><a hzef="#">联系</a></li>
</ul>
```

使用 id 标识页面上的元素时，id 名必须是唯一的。id 可以用来标识持久的结构性元素，如主导航或内容区域；还可以用来标识一次性元素，如某个链接或表单元素。

在整个网站上，id 名应该应用于语义相似的元素以避免混淆。例如，如果联系人表单和联系人详细信息在不同的页面上，那么可以给它们分配同样的 id 名 contact，但是如果在外部样式表中给它们定义样式，就会遇到问题，因此使用不同的 id 名（如 contact_form 和 contact_details）就会简单得多。

与 id 不同，同一个 class 可以应用于页面上任意数量的元素，因此 class 非常适合标识样式相同的对象。例如，设计一个新闻页面，其中包含每条新闻的日期。此时不必给每个日期分配不同的 id，而是可以给所有日期分配类名 date。





提示：id 和 class 的名称一定要保持语义性，并与表现方式无关。例如，可以给导航元素分配 id 名为 right_nav，因为希望它出现在右边。但是，如果以后将它的位置改到左边，那么 CSS 和 HTML 就会发生歧义。所以，将这个元素命名为 sub_nav 或 nav_main 更合适。这种名称解释就不再涉及如何表现它。对于 class 名称也是如此。例如，如果定义所有错误消息以红色显示，不要使用类名 red，而应该选择更有意义的名称，如 error 或 feedback。

注意：class 和 id 名称需要区分大小写，虽然 CSS 不区分大小写，但是在标签中是否区分大小写取决于 HTML 文档类型。如果使用 XHTML 严谨型文档，那么 class 和 id 名是区分大小写的。最好的方式是保持一致的命名约定，如果在 HTML 中使用驼峰命名法，那么在 CSS 中也采用这种形式。

【示例 2】在实际设计中，class 被广泛使用，这就容易产生滥用现象。例如，很多初学者在所有的元素上添加类，以便更方便地控制它们。这种现象被称为“多类症”，在某种程度上，这和使用基于表格的布局一样糟糕，因为它在文档中添加了无意义的代码。

```
<h1 class="newsHead">标题新闻</h1>
<p class="newsText">新闻内容</p>
<p>......</p>
<p class="newsText"><a href="news.php" class="newsLink">更多</a></p>
```

【示例 3】在示例 2 中，每个元素都使用一个与新闻相关的类名进行标识。这使新闻标题和正文可以采用与页面其他部分不同的样式。但是，不需要用这么多类来区分每个元素，可以将新闻条目放在一个包含框中，并加上类名 news，从而标识整个新闻条目。然后，可以使用包含框选择器识别新闻标题或文本，代码如下。

```
<div class="news">
    <h1>标题新闻</h1>
    <p>新闻内容</p>
    <p>......</p>
    <p><a href="news.php">更多</a></p>
</div>
```

以这种方式删除不必要的类有助于简化代码，使页面更简洁。过度依赖类名是不必要的，我们只需要在不适合使用 id 的情况下对元素应用类，而且尽可能少使用类。实际上，创建大多数文档常常只需要添加几个类。如果初学者发现自己添加了许多类，那么这很可能意味着自己创建的 HTML 文档结构有问题。

9.1.4　设置文档类型



在网页文档的第一行代码中，一般都要使用<doctype>标签定义文档的类型，示例如下。





Note

☑　定义 HTML5 类型文档

```
<!doctype html>
```

☑　定义 XHTML 1.0 过渡型文档

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

☑　定义 HTML 4.01 严谨型文档

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
```

网页文档类型众多，主要根据HTML版本号细分。常用类型包括HTML 4.01、XHTML 1.0、HTML5 和 XHTML Mobile 1.0，其中 HTML 4.01 和 XHTML 1.0 又分为过渡型和严谨型两种。

浏览器根据<doctype>标签是否存在和设置的 DTD 来选择要使用的表现方法。如果 XHTML 文档包含形式完整的 DOCTYPE，那么它一般以标准模式表现。

提示： DTD（文档类型定义）是一组机器可读的规则，它们定义 XML 或 HTML 的特定版本中允许有什么，不允许有什么。在解析网页时，浏览器将使用这些规则检查页面的有效性，并且采取相应的措施。浏览器通过分析页面的 DOCTYPE 声明来了解要使用哪个 DTD，因此知道要使用 HTML 的哪个版本。


视 频 讲 解

9.1.5　认识显示模式

浏览器为了实现对标准网页和传统网页的兼容，分别制订了几套网页显示方案，这些方案就是浏览器的显示模式。浏览器能够根据网页文档类型来决定选择哪套显示模式对网页进行解析。

☑　IE 浏览器支持两种显示模式：标准模式和怪异模式。在标准模式中，浏览器会根据 W3C 制定的标准来显示页面；而在怪异模式中，将以 IE 5 显示页面的方式来呈现网页，以保证与过去非标准网页的兼容。

☑　Firefox 支持 3 种显示模式：标准模式、几乎标准的模式和怪异模式。其中几乎标准的模式对应于 IE 和 Opera 的标准模式，该模式除了在处理表格的方式方面有一些细微差异外，与标准模式基本相同。

☑　Opera 支持与 IE 相同的显示模式。但是在 Opera 9 版本中，怪异模式不再兼容 IE 5 盒模型解析方式。

【示例】为了理解标准模式和怪异模式，本示例比较了浏览器显示模式的工作方式。

第 1 步，新建文档，完整地输入下面的网页代码。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3c.org/TR/1999/REC-html401-19991224/loose.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```






```
<title>标准模式</title>
<style type=text/css>
div {
 border:solid 50px red;
 padding:50px;
 background:#ffccff;
 width:200px;
 height:100px;
}
</style>
</head>
<body>
<div>标准显示盒模型</div>
</body>
</html>
```

第 2 步，新建文档，完整地输入下面的网页代码。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>怪异模式</title>
<style type=text/css>
div {
	border:solid 50px red;
	padding:50px;
	background:#ffccff;
	width:200px;
	height:100px;
}
</style>
</head>
<body>
<div>怪异显示盒模型</div>
</body>
</html>
```

第 3 步，保存文档，在 IE 浏览器中预览上面创建的两个文档，效果如图 9.2 和图 9.3 所示。









图 9.2　标准模式显示效果

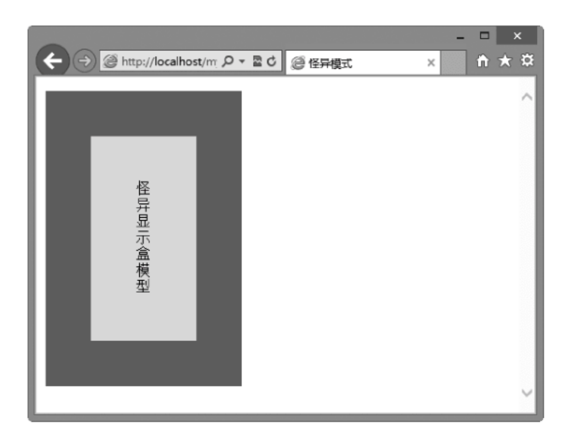


图 9.3　怪异模式显示效果

可以看到，当网页的文档类型被声明为 HTML 4.01 过渡型时，网页将按标准模式显示，页面显示的盒模型将遵循 W3C 制定的标准进行解析。

注意： 对于 HTML 1.01 文档，包含严格 DTD 的 DOCTYPE 常常导致页面以标准模式解析，包含过渡 DTD 和 URI 的 DOCTYPE 也导致页面以标准模式解析。但是有过渡 DTD 而没有 URI 会导致页面以怪异模式表现。DOCTYPE 不存在或形式不正确会导致 HTML 和 XHTML 文档以怪异模式表现。例如，定义下面几种文档类型，IE 浏览器会以怪异模式显示网页。

☑ 没有提供文档类型的版本

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML//EN" "http://www.w3.org/TR/html/loose.dtd">
```

☑ HTML 2.0 版本

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
```

☑ HTML 3.0 版本

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3.0//EN//">
```

☑ HTML 3.2 版本

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
```

9.2　CSS 布局基础

学习 CSS 布局，应该先了解 CSS 盒模型，本节将介绍 CSS 盒模型相关的构成要素和应用技巧。









9.2.1　CSS 盒模型结构

在网页设计中，常会听到这些名词概念：内容（content）、填充（补白，内边距，padding）、边框（border）、边界（外边距，margin）。日常生活中的盒子就是能装东西的一种箱子，也具有这些属性，所以把这些名词抽象为盒子模型概念。它具有如下特性。

☑　每个盒子都有边界、边框、填充和内容 4 个属性。

☑　每个属性都包括上、右、下、左 4 个部分。属性的 4 个部分可同时设置，也可分别设置。

内容（content）就是盒子里装的东西，而填充（padding）就是怕盒子里装的东西损坏而填加的泡沫或者其他抗震辅料，边框（border）就是盒子本身了，至于边界（margin），则说明盒子摆放的时候不能全部堆在一起，要留一定空隙保持通风，同时也为了方便取出。

在网页中，内容常指文字、图片等信息或元素，也可以是小盒子（嵌套结构），与现实生活中盒子不同的是，现实生活中的东西一般不能大于盒子，否则盒子会被撑坏，而 CSS 盒子具有弹性，里面的东西大过盒子最多把盒子撑大，不会损坏盒子。网页元素与盒子之间的关系如图 9.4 所示。

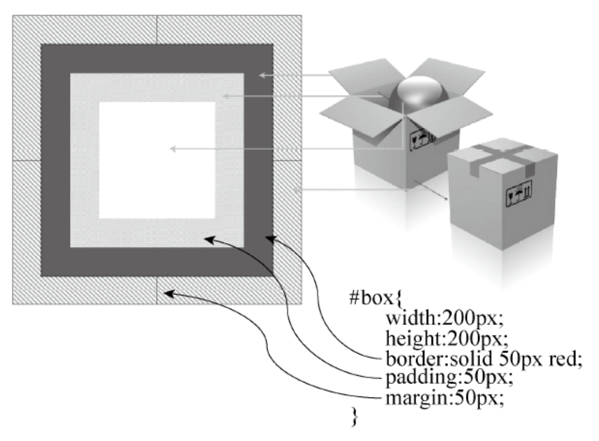


图 9.4　CSS 元素与盒子结构关系示意图

网页中任何元素都可以视为一个盒子，所有盒模型就是页面元素的基本模型结构。从外到里，盒模型包括边界、边框、补白和内容 4 大区域，结构示意图如图 9.5 所示。

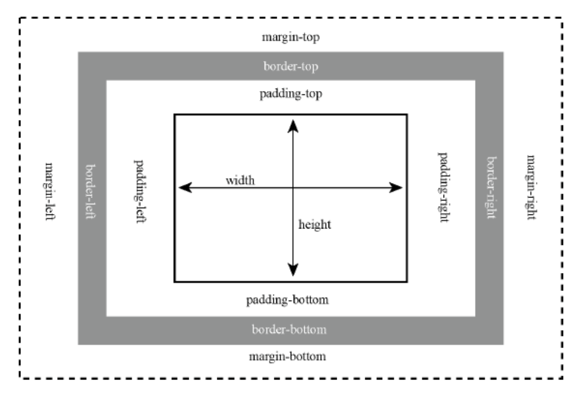


图 9.5　盒模型中各个属性的空间位置关系




Note


视频讲解

9.2.2 盒子大小

CSS 盒子模型使用 width（宽）和 height（高）定义内容区域的大小。

【示例】本示例定义两个并列显示的 div 元素，设置每个 width 为 50%，显示效果如图 9.6 所示。

```
<style type="text/css">
div {/*定义 div 元素公共属性 */
    float: left;                    /*向左浮动，实现并列显示*/
    text-align: center;             /* 定义 div 内的字体居中显示 */
    height: 240px;                  /* 定义高度*/
}
#left { background-color:rgba(0,0,255,0.3); width: 50%;}
#right {background-color:rgba(255,0,0,0.3); width: 50%;}
</style>

<div id="left">左栏</div>
<div id="right">右栏</div>
```



图 9.6　定义元素的大小

提示：内容区域包括宽度（width）、高度（height）和背景（background），实际上背景能够延伸到补白区域，有些浏览器中背景图像甚至延伸到边框内。所以对于一个 CSS 盒模型来说，它的实际宽度或高度就等于内容区域的宽（width）和高（height）加上二倍的边界、边框和补白之和。

W = width（content）+ 2 * (border + padding + margin)

H = height（content）+ 2 * (border + padding + margin)

在 IE 怪异解析模式下（其他部分浏览器也支持怪异模式，如 Firefox 和 Opera 等），CSS 的 width 属性表示内容区域、补白和边框宽度之和，而 height 属性表示内容区域、补白和边框高度之和，所以盒模型的实际大小就为：







```
W = width + 2 * margin
H = height + 2 * margin
```

用户也可以通过下面 4 个属性灵活控制 CSS 盒模型的大小，这些属性在网页弹性布局中非常有用。

- ☑ min-width：设置对象的最小宽度。
- ☑ max-width：设置对象的最大宽度。
- ☑ min-height：设置对象的最小高度。
- ☑ max-height：设置对象的最大高度。

9.2.3　盒子边框

边框样式由 CSS 的 border 属性负责定义，border 包括 3 个子属性：border-style（边框样式）、border-color（边框颜色）和 border-width（边框宽度）。

1. 定义宽度

定义边框的宽度有多种方法，简单说明如下。

方法 1，直接在属性后面指定宽度值，代码如下。

```
<style type="text/css">
border-bottom-width:12px;          /*定义元素的底边框宽度为 12px*/
border-top-width:0.2em;            /*定义顶部边框宽度为元素内字体大小的 0.2 倍*/
</style>
```

方法 2，使用关键字，如 thin、medium 和 thick。thick 比 medium 宽，而 medium 比 thin 宽。不同浏览器对此解析的宽度值也不同，有的解析为 5px、3px、2px，有的解析为 3px、2px、1px。

方法 3，单独为元素的某条边设置宽度，可以使用 border-top-width（顶边框宽度）、border-right-width（右边框宽度）、border- bottom-width（底边框宽度）和 border-left-width（左边框宽度）。

方法 4，使用 border-width 属性快速定义边框宽度，例如：

```
<style type="text/css">
border-width:2px;               /*定义四边都为 2px*/
border-width:2px 4px;           /*定义上下边为 2px，左右边为 4px*/
border-width:2px 4px 6px;       /*定义上边为 2px，左右边为 4px，底边为 6px*/
border-width:2px 4px 6px 8px;   /*定义上边为 2px，右边为 4px，底边为 6px，左边为 8px*/
</style>
```

提示： 当定义边框宽度时，必须要定义边框的显示样式。由于边框默认样式为 none，即不显示，所以仅设置边框的宽度，由于样式不存在，边框宽度也自动被清除为 0。

2. 定义样式

边框样式是边框显示的基础，CSS2 提供了下面几种边框样式。



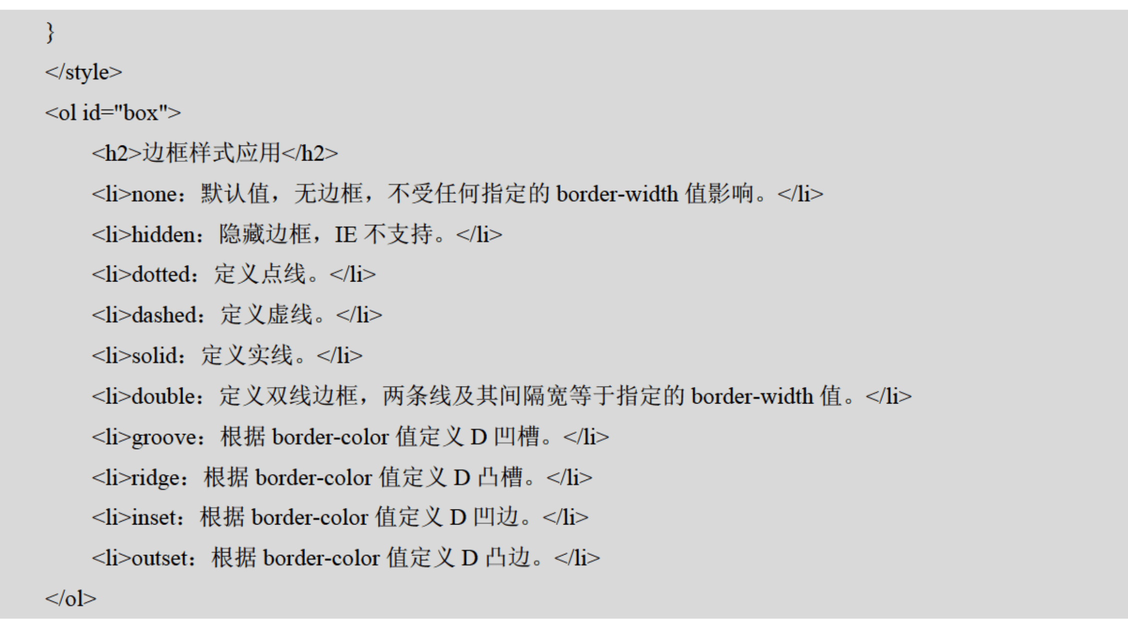

Note

☑ none：默认值，无边框，不受任何指定的 border-width 值影响。
☑ hidden：隐藏边框，IE 不支持。
☑ dotted：定义边框为点线。
☑ dashed：定义边框为虚线。
☑ solid：定义边框为实线。
☑ double：定义边框为双线边框，两条线及其间隔宽度之和等于指定的 border-width 值。
☑ groove：根据 border-color 值定义 3D 凹槽。
☑ ridge：根据 border-color 值定义 3D 凸槽。
☑ inset：根据 border-color 值定义 3D 凹边。
☑ outset：根据 border-color 值定义 3D 凸边。

【示例】本示例使用 border 设计列表框样式，定义列表项显示下划线，效果如图 9.7 所示。

```
<style type="text/css">
#box {/*<定义信纸的外框>*/
    width: 500px;
    height: 400px;
    padding: 8px 24px;
    margin: 6px;
    border-style: outset;                /* 定义信纸边框为 3D 凸边效果 */
    border-width: 4px;                   /* 定义信纸边框宽度 */
    border-color: #aaa;                  /* 定义信纸边框颜色 */
    font-size: 14px;
    color: #D02090;
    list-style-position: inside;         /* 定义列表符号在内部显示 */
}
#box h2 {/*<定义标题格式>*/
    padding-bottom: 12px;
    border-bottom-style: double;         /* 定义标题底边框为双线显示 */
    border-bottom-width: 6px;            /* 定义标题底边框宽度 */
    border-bottom-color: #999;           /* 定义标题底边框颜色 */
    text-align: center;
    color: #000000;
}
#box li {
    padding: 6px 0;                      /* 增加列表项之间的间距 */
    border-bottom-style: dotted;         /* 定义列表项底边框为点线显示 */
    border-bottom-width: 1px;            /* 定义列表项底边框宽度 */
    border-bottom-color: #66CC66;        /* 定义列表项底边框颜色 */
```





```
}
</style>
<ol id="box">
    <h2>边框样式应用</h2>
    <li>none：默认值，无边框，不受任何指定的 border-width 值影响。</li>
    <li>hidden：隐藏边框，IE 不支持。</li>
    <li>dotted：定义点线。</li>
    <li>dashed：定义虚线。</li>
    <li>solid：定义实线。</li>
    <li>double：定义双线边框，两条线及其间隔宽等于指定的 border-width 值。</li>
    <li>groove：根据 border-color 值定义 D 凹槽。</li>
    <li>ridge：根据 border-color 值定义 D 凸槽。</li>
    <li>inset：根据 border-color 值定义 D 凹边。</li>
    <li>outset：根据 border-color 值定义 D 凸边。</li>
</ol>
```



IE 预览效果



Firefox 预览效果

图 9.7　边框样式比较

在 IE 和 Firefox 浏览器中分别进行预览，则效果存在细微区别，说明不同浏览器在解析相同的样式代码时显示效果也不完全相同。

提示： 在默认状态下，边框的宽度为 medium（中型），这是一个相对宽度，一般为 2～3 像素。边框默认样式为 none，即隐藏边框显示。边框默认颜色为前景色，即元素中包含文本的颜色，如果没有文字，则将继承上级元素所包含的文本颜色。

9.2.4　盒子边界



元素的边距由 CSS 的 margin 属性控制，margin 定义了元素与其他相邻元素的距离。由 margin 属性又派生出 4 个子属性：margin-top（顶部边界）、margin-right（右侧边界）、margin-bottom（底部



Note

边界）和 margin-left（左侧边界），这些属性控制元素在不同方位上与其他元素的间距。

【示例】在本示例中，设置 4 个盒子的外边界变化，通过不同方向上外边界的设置，设计出一个梯状效果，如图 9.8 所示。通过本例演示，用户能够体验到边界可以自由设置，且各边边界不会相互影响。

```
<style type="text/css">
div { /* <div>标签的默认样式 */
    height: 20px;                                          /* 统一高度 */
    border: solid 1px red;                                 /* 统一边框样式 */
}
#box4 {/* 第 4 个盒子样式 */
    margin-top: 2px;                                       /* 顶部边界大小 */
    margin-right: 1em;                                     /* 右侧边界大小 */
    margin-left: 1em;                                      /* 左侧边界大小 */
}
#box3 {margin-top: 4px; margin-right: 4em; margin-left: 4em;}     /* 第 3 个盒子样式 */
#box2 {margin-top: 8px; margin-right: 8em; margin-left: 8em;}     /* 第 2 个盒子样式 */
#box1 {margin-top: 16px; margin-right: 12em; margin-left: 12em;}  /* 第 1 个盒子样式 */
</style>
<div id="box1"></div>
<div id="box2"></div>
<div id="box3"></div>
<div id="box4"></div>
```

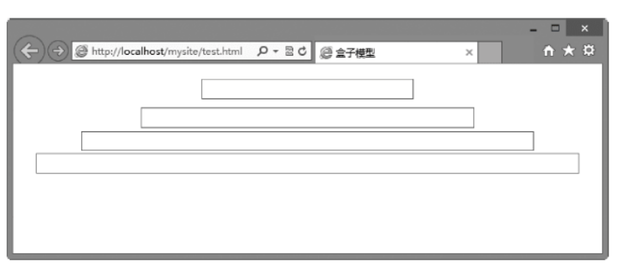
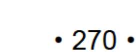


图 9.8　盒模型不同方向上边界的设置效果



提示：为了提高代码编写效率，CSS 提供了边界定义的简写方式。具体说明如下。

- 如果 4 个边界相同，则直接使用 margin 属性定义，为 margin 设置一个值即可。
- 如果 4 个边界不相同，则可以在 margin 属性中定义 4 个值，4 个值用空格进行分隔，代表边的顺序是顶部、右侧、底部和左侧，即从顶部开始按顺时针方向进行设置。代码如下。

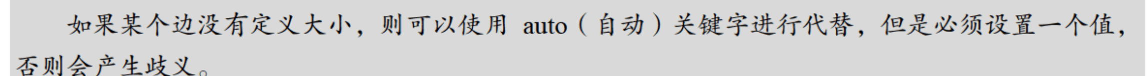
```
margint:top right bottom left;
```





这样就能够加速代码输入速度。例如，针对上面示例中的样式可以简写成如下样式。

```
#box4 { margin:1px 1em auto 1em; }
#box3 { margin:4px 4em auto 4em; }
#box2 { margin:8px 8em auto 8em; }
#box1 { margin:12px 12em auto 12em; }
```



如果某个边没有定义大小，则可以使用 auto（自动）关键字进行代替，但是必须设置一个值，否则会产生歧义。

- 如果上下边界不同，左右边界相同，则可以使用 3 个值进行代替，因此可以简写成如下样式。

```
margint:top right bottom;
```

例如，针对上面的示例代码，可以继续简写成如下样式。

```
#box4 { margin:1px 1em auto; }
#box3 { margin:4px 4em auto; }
#box2 { margin:8px 8em auto; }
#box1 { margin:12px 12em auto; }
```

因为左右边界相同，所以就不用再考虑，合并在一起即可。

- 如果上下边界相同，左右边界相同，则直接使用两个值进行代替：第 1 个值表示上下边界，第 2 个值表示左右边界。例如，下面样式定义了段落文本的上下边界为 12 像素，而左右边界为 24 像素。

```
p{ margin:12px 24px;}
```

注意：margin 可以取负值，这样就能够强迫元素偏移原来位置，实现相对定位功能，利用 margin 功能，可以设计复杂的页面布局效果。

9.2.5 盒子补白



补白用来调整元素包含的内容与元素边框的距离，由 CSS 的 padding 属性负责定义。从功能上讲，补白不会影响元素的大小，但是由于在布局中补白同样占据空间，所以在布局时应考虑补白对于布局的影响。如果在没有明确定义元素的宽度和高度的情况下，使用补白来调整元素内容的显示位置要比边界更加安全、可靠。

padding 与 margin 属性一样，也包括 4 个子属性：padding-top、padding-right、padding-bottom 和 padding-left，这些子属性分别定义 4 边的补白大小。例如：

```
padding:2px;/*定义元素四周补白为 2px*/
padding:2px 4px;/*定义上下补白为 2px，左右补白为 4px*/
padding:2px 4px 6px;/*定义上补白为 2px，左右补白为 4px，下补白为 6px*/
padding:2px 4px 6px 8px; /*定义上补白为 2px，右补白为 4px，下补白为 6px，左补白为 8px*/
```






```
padding-top:2px;/*定义元素上补白为 2px*/
padding-right:2em;/*定义右补白为元素字体的两倍*/
padding-bottom:2%;/*定义下补白为父元素宽度的 2%*/
padding-left:auto; /*定义左补白为自动*/
```

与边界不同，补白取值不可以为负。补白和边界一样都是透明的，当设置元素的背景色或边框后，才能感觉到补白的存在。

【示例】本示例设计的是导航列表项水平显示，通过补白调整列表项的大小，如图 9.9 所示。

```
<style type="text/css">
ul {/*清除列表样式*/
    margin: 0;   /*清除 IE 列表缩进*/
    padding: 0;  /*清除非 IE 列表缩进*/
    list-style-type: none;      /*清除列表样式*/
}
#nav {width: 100%;height: 32px;} /*定义列表框宽和高*/
#nav li {/*定义列表项样式*/
    float: left;    /*浮动列表项*/
    width: 9%;  /*定义百分比宽度*/
    padding: 0 5%;     /*定义百分比补白*/
    margin: 0 2px;     /*定义列表项间隔*/
    background: #def;        /*定义列表项背景色*/
    font-size: 16px;
    line-height: 32px;  /*垂直居中*/
    text-align: center;  /*平行居中*/
}
</style>
<ul id="nav">
    <li>美 丽 说</li>
    <li>聚美优品</li>
    <li>唯 品 会</li>
    <li>蘑 菇 街</li>
    <li>1 号 店</li>
</ul>
```



图 9.9　IE 下预览效果





注意：补白与边界、边框一样都是可选的，并不是每个元素都必须全部设置，如果不设置这些属性，CSS 默认其值为 0。但是很多元素已经被浏览器预定了特定样式，如 body、p、h1～h6、ul 等，这些预定义样式主要包括补白和边界的属性设置。当然，也可以重置 margin 和 padding 为 0，清除其中的预定义样式。为了快速开发，可以在页面设计之初，用通配选择符清除所有元素的补白和边界样式，代码如下

```
* {/*[清除所有元素的预定义样式]*/
    margin:0; /*清除边界值*/
    padding:0; /*清除补白值*/
}
```

9.2.6　认识显示类型



在 CSS 中，可以使用 display 属性来改变元素的显示类型。在 CSS 2.1 中，display 属性共有 18 个选项值，详细说明如下。

- ☑ block：块状显示，在元素后面添加换行符，也就是说其他元素不能在其后面并列显示。
- ☑ none：隐藏显示，这与 visibility:hidden;声明不同，display:none;声明不会为被隐藏的元素保留位置。
- ☑ inline：行内显示，在元素后面删除换行符，多个元素可以在一行内并列显示。
- ☑ inline-block：行内显示，以块状显示，其他行内元素可以同行显示。
- ☑ compact：紧凑的块状显示或基于内容之上的行内显示。
- ☑ marker：在容器对象之前或之后显示，该属性值必须与:after 和:before 伪元素一起使用。
- ☑ inline-table：具有行内特征的表格显示。
- ☑ list-item：具有块状特征的列表项目显示，并可以添加可选项目标志。
- ☑ run-in：块状显示或基于内容之上的行内显示。
- ☑ table：具有块状特征的表格显示。
- ☑ table-caption：表格标题显示。
- ☑ table-cell：表格单元格显示。
- ☑ table-column：表格列显示。
- ☑ table-column-group：表格列组显示。
- ☑ table-header-group：表格标题组显示。
- ☑ table-footer-group ：表格页脚组显示。
- ☑ table-row：表格行显示。
- ☑ table-row-group：表格行组显示。

CSS3 新增了 box 显示类型，关于该技术话题请参阅下一章内容。

在常规网页设计中，CSS 把标签分为两种基本显示形态：Block（块状）和 Inline（行内）。块状元素的宽度一般为 100%，占据一行，即使宽度不为 100%，块状元素也始终占据一行。常用块状元素如表 9.1 所示。





表 9.1 符合标准的常用块状元素

块 状 元 素	说　　明
address	表示特定信息，如地址、签名、作者、文档信息。一般显示为斜体效果
blockquote	表示文本中的一段引用语。一般为缩进显示
div	表示通用定位包含框，没有明确的语义
dl	表示定义列表
fieldset	表示字段集，显示为一个方框，用来包含文本和其他元素
form	说明所包含的控件是某个表单的组成部分
h1-h6	表示标题，其中 h1 表示一级标题，字号最大，h6 表示最小级别标题，字号最小
hr	画一条横线
noframes	包含对于那些不支持 FrameSet 元素的浏览器使用的 HTML
noscript	指定在不支持脚本的浏览器中显示的 HTML
ol	编制有序列表
p	表示一个段落
pre	以固定宽度字体显示文本，保留代码中的空格和回车符
table	表示将所含内容组织成含有行和列的表格形式
ul	表示不排序的项目列表
li	表示列表中的一个项目
legend	在 FieldSet 元素绘制的方框内插入一个标题

行内元素没有固定的大小，定义它的 width 和 height 属性无效。行内元素可以在行内自由流动，但可以定义边界、补白、边框和背景，它显示的高度和宽度只能够根据所包含内容的高度和宽度来确定。常用行内元素如表 9.2 所示。

表 9.2 符合标准的常用行内元素

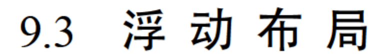
行 内 元 素	说　　明
a	表示超链接
abbr	标注内部文本为缩写，用 title 属性标示缩写的全称，在非 IE 浏览器中会以下点划线显示，IE 不支持
acronym	表示取首字母的缩写词，一般显示为粗体，部分浏览器支持
b	指定文本以粗体显示
bdo	用于控制包含文本的阅读顺序，如<bdo dir="rtl">this fragment is in english</bdo>，浏览器会从右到左显示文本
big	指定所含文本要以比当前字体稍大的字体显示
br	插入一个换行符
button	指定一个容器，可以包含文本，显示为一个按钮
cite	表示引文，以斜体显示
code	表示代码范例，以等宽字体显示





续表

行内元素	说　明
dfn	表示术语，以斜体显示
em	表示强调文本，以斜体显示
i	指定文本以斜体显示
img	插入图像或视频片断
input	创建各种表单输入控件
kbd	以定宽字体显示文本
label	为页面上的其他元素指定标签
map	包含客户端图像映射的坐标数据
object	插入对象
q	分离文本中的引语
samp	表示代码范例
script	指定由脚本引擎解释的页面中的脚本
select	表示一个列表框或者一个下拉框
small	指定内含文本要以比当前字体稍小的字体显示
span	指定内嵌文本容器
strike	带删除线显示文本
strong,	以粗体显示文本
sub	说明内含文本要以下标的形式显示，比当前字体稍小
sup	说明内含文本要以上标的形式显示，比当前字体稍小
textarea	多行文本输入控件
tt,	以固定宽度字体显示文本
var	定义程序变量，通常以斜体显示

9.3　浮 动 布 局

浮动布局不同于流动模式，它能够让对象脱离 HTML 结构，在包含框内向左侧或右侧浮动显示，但是浮动元素不能脱离文档流，它依然受文档流的影响。

9.3.1　定义浮动显示



在默认情况下，任何元素都不具有浮动特性，可以使用 CSS 的 float 属性定义元素向左或向右浮动，基本用法如下。

```
float: none | left | right
```

其中，none 表示消除浮动，left 表示元素向左浮动，right 表示元素向右浮动，默认值为 none。



浮动布局模型具有下面几个特征。

第一，浮动元素以块状显示，可以定义 width 和 height 属性。



【示例 1】本示例为两个 span 元素定义高和宽属性，然后让其中一个 span 元素浮动显示，从而比较它们的显示效果，如图 9.10 所示。

```
<style type="text/css">
span {/*定义行内元素 span 的显示属性*/
 width:400px;                  /*定义宽为 400 像素*/
 height:200px;                 /*定义高为 200 像素*/
 border:solid red 1px;
}
#inline img {width:100px; }    /*定义行内元素内的图片宽为 100 像素*/
#float {float:right;}          /*为第 2 个行内元素 span 定义浮动显示*/
</style>
<span id="inline">行内元素流动显示
 <img src="images/1.jpg" alt="流动的图片" />
</span>
<span id="float">行内元素浮动显示</span>
```



行内元素浮动显示 1



行内元素浮动显示 2

图 9.10　浮动显示与流动显示比较

通过图 9.10 可以看到当第 2 个 span 元素被定义为浮动之后，该元素自动以块状显示，因此为 span 元素定义的高和宽属性值有效。而第 1 个元素由于是行内元素且没有浮动显示，所以定义的宽和高无效，所看到的红色边框仅包裹在行内元素的外边。

浮动元素应该明确定义大小。如果浮动元素没有定义宽度和高度，它会自动收缩到仅能包住内容为止。例如，如果浮动元素内部包含一张图片，则浮动元素将与图片一样宽，如果是包含的文本，则浮动元素将与最长文本行一样宽。而当块状元素没有定义宽度时，则会自动显示为 100%。

第二，浮动元素与流动元素可以混合使用，不会重叠，都遵循先上后下的显示规则，都受到文档流影响。但浮动元素能够改变相邻元素的显示位置，可以向左或向右并列显示。

与普通元素一样，浮动元素始终位于包含元素内，不会脱离包含框，这与定位元素不同。





【示例 2】本示例以示例 1 为基础，添加一个包含框，可以看到第 2 个 span 元素靠近包含元素 div 的右边框浮动，而不再是 body 元素的右边框，如图 9.11 所示。

```
<div id="contain">
    <span id="inline">行内元素流动显示
        <img src="images/1.jpg" alt="流动的图片" />
    </span>
    <span id="float">行内元素浮动显示</span>
</div>
```



图 9.11　浮动元素始终位于包含元素

第三，浮动元素仅能改变水平显示方式，不能改变垂直显示方式，并且依然受文档流影响。流动元素总会以流动的形式环绕浮动元素左右显示。

浮动元素不会强迫前面的流动元素环绕其周围流动，而总是在上邻流动元素的下一行浮动显示。浮动元素不会覆盖其他元素，也不会挤占其他元素的位置。

第四，浮动元素可以并列显示，如果包含框宽度不够，则会错行显示。

【示例 3】本示例模拟设计了 3 个并列显示的栏目，通过 float 定义左、中、右 3 栏并列显示。代码如下，效果如图 9.12 所示。

```
<style type="text/css">
body {padding: 0; margin: 0; text-align: center;}
#main {/*定义网页包含框样式*/
    width: 400px;
    margin: auto;
    padding: 4px;
    line-height: 160px;
    color: #fff;
    font-size: 20px;
    border: solid 2px red;
}
```





```
#main div {float: left;height: 160px;}          /*定义 3 个并列栏目向左浮动显示*/
#left {width: 100px;background: red;}           *定义左侧栏目样式*/
#middle {width: 200px;background: blue;}        /*定义中间栏目样式*/
#right {width: 100px; background: green;}       /*定义右侧栏目样式*/
.clear { clear: both; }
</style>
<div id="main">
    <div id="left">左侧栏目</div>
    <div id="middle">中间栏目</div>
    <div id="right">右侧栏目</div>
    <br class="clear" />
</div>
```



图 9.12　并列浮动显示

浮动布局可以设计多栏并列显示效果，但也容易错行，如果浏览器窗口发生变化或者浮动包含框不固定，则会出现错行浮动显示问题，导致破坏并列布局效果。


视 频 讲 解

9.3.2　清除浮动

使用 CSS 的 clear 属性可以清除浮动，定义与浮动相邻的元素在必要的情况下换行显示，这样可以控制浮动元素挤在一行内显示。clear 属性取值包括以下 4 个。

☑　left：清除左边的浮动元素，如果左边存在浮动元素，则当前元素会换行显示。

☑　right：清除右边的浮动元素，如果右边存在浮动元素，则当前元素会换行显示。

☑　both：清除左右两边的浮动元素，不管哪边存在浮动对象，则当前元素都会换行显示。

☑　none：默认值，允许两边都可以存在浮动元素，当前元素不会主动换行显示。

【示例】本示例设计一个 3 行 3 列的页面结构，定义中间 3 栏浮动显示。代码如下，显示效果如图 9.13 所示。

```
<style type="text/css">
div {
    border: solid 1px red;                  /* 增加边框，以方便观察 */
    height: 50px;                           /* 固定高度，以方便比较 */
```





```
    }
#left, #middle, #right {
    float: left;                    /* 定义中间 3 栏向左浮动 */
    width: 33%;                     /* 定义中间 3 栏等宽 */
}
</style>
<div id="header">头部信息</div>
<div id="left">左栏信息</div>
<div id="middle">中栏信息</div>
<div id="right">右栏信息</div>
<div id="footer">脚部信息</div>
```

但是如果设置左栏高度大于中栏和右栏高度，则发现脚部信息栏上移并环绕在左栏右侧，代码如下，显示效果如图 9.14 所示。

```
#left {height:100px; }              /* 定义左栏高出中栏和右栏 */
```

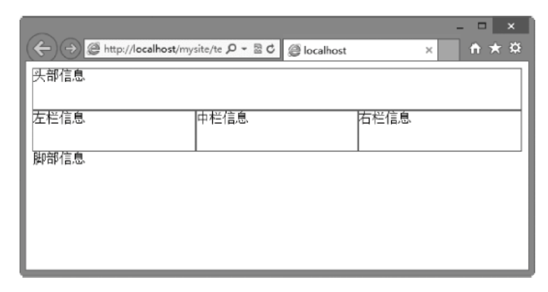


图 9.13　IE 6 中浮动布局效果

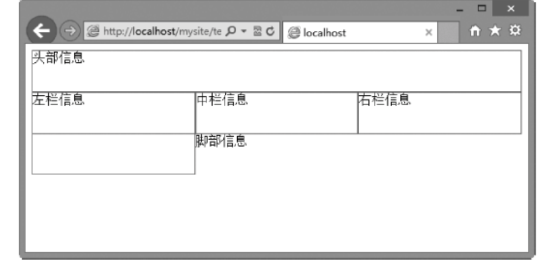
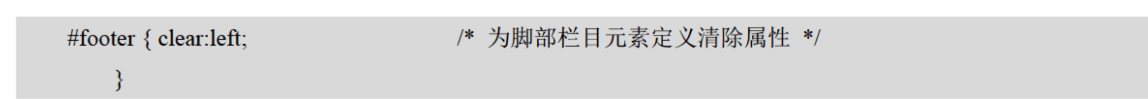


图 9.14　调整部分栏目高度后发生的错位现象

这时 clear 属性就可以派上用场了，为<div id="footer">元素定义一个清除样式，代码如下。



```
#footer { clear:left;               /* 为脚部栏目元素定义清除属性 */
    }
```

在浏览器中预览，则又恢复到预设的 3 行 3 列布局效果，如图 9.15 所示。

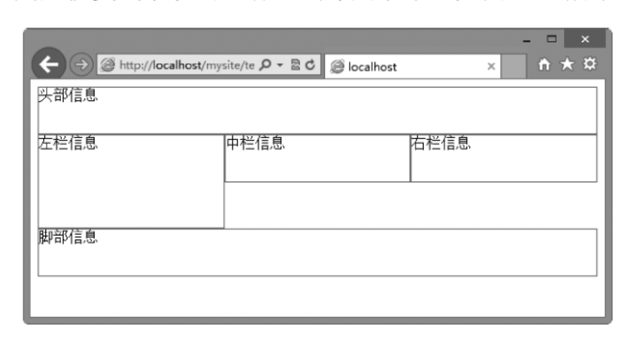


图 9.15　清除浮动元素错行显示





视频讲解

提示：clear 属性是专门针对 float 属性而设计的，因此仅能够对左右两侧浮动元素有效，对于非浮动元素是无效的。

这里所谓的清除，不是清除浮动元素，而是清除自身，通俗地说就是不允许当前元素与浮动元素并列显示。如果左右两侧存在浮动元素，则当前元素把自己清除到下一行显示，而不是把前面的浮动元素清除到下一行显示或者清除到上一行显示。

9.3.3 浮动嵌套

浮动元素可以相互嵌套，嵌套规律与流动元素嵌套相同。浮动的包含元素总会自动调整自身高度和宽度以实现对浮动子元素的包含。

【示例 1】新建文档，构建两个简单的嵌套块，然后强制它们浮动显示，并定义子元素的高度和宽度，使其显示为一定大小的区域，这时会发现父元素会自动调整自身大小来包含子元素。代码如下，效果如图 9.16 所示。

```
<style type="text/css">
.wrap { border: solid 2px red; float: left; margin:4px;}
.sub { width: 200px; height: 200px; float: left; background: blue; }
</style>
<div class="wrap">
    <div class="sub"></div>
</div>
<span class="wrap">
    <span class="sub"></span>
</span>
```

提示：如果包含元素定义了高度和宽度，则它就不会随子元素的大小而自动调整自身显示区域来适应子元素的显示，如图 9.17 所示。注意，在 IE 6 及更低版本的浏览器中包含框仍然能够自动调整自身大小来适应子元素的显示大小，不过在 IE 7 版本中，微软纠正了这个不符合标准的显示方法。

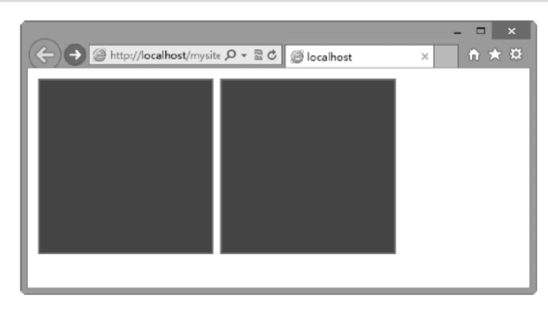


图 9.16　浮动嵌套

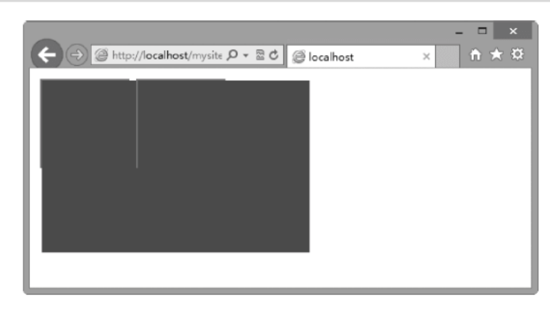


图 9.17　浮动嵌套存在问题





【示例 2】如果把浮动元素嵌入流动元素内，则父元素不能自适应子浮动元素的高度。代码如下，效果如图 9.18 所示。

```
<style type="text/css">
#contain { background: #FF99FF; }                /*包含元素*/
span {float: left; width: 200px; height: 100px; }    /*定义共同属性*/
/*内嵌浮动对象样式 */
#span1 { border: solid blue 10px; }
#span2 { border: solid red 10px; }
</style>
<div id="contain">
    <span id="span1">span 元素浮动</span>
    <span id="span2">span 元素浮动</span>
</div>
```

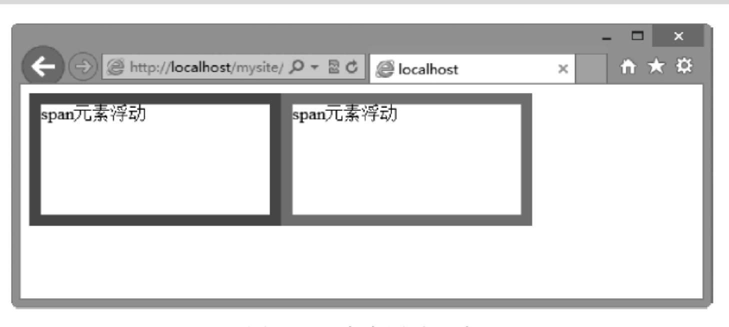


图 9.18　嵌套浮动元素

在图 9.18 中可以看到包含元素 div 并没有显示。原因就是包含元素没有适应子元素的高度，而是根据自身定义的属性以独立的形式显示。所以，在应用混合嵌套时，要预测到浮动与流动混合布局时会出现的各种怪现象，并积极做好兼容处理。

解决方法：可以在包含元素内的最后一行添加一个清除元素，强制撑开包含元素，使其包含浮动元素，代码如下，显示结果如图 9.19 所示。

```
<style type="text/css">
#contain { background: #FF99FF; }                    /*包含元素*/
span {float: left; width: 200px; height: 100px; }        /*定义共同属性*/
/*内嵌浮动对象样式 */
#span1 { border: solid blue 10px; }
#span2 { border: solid red 10px; }
.clear {clear:both; }                                /*定义清除类*/
</style>
<div id="contain">
    <span id="span1">span 元素浮动</span>
```





Note

```
        <span id="span2">span 元素浮动</span>
        <div class="clear"></div><!--增加一个清除元素-->
</div>
```



图 9.19　正确显示效果



视 频 讲 解

9.3.4　混合浮动布局

浮动布局模型相比流动布局模型要复杂很多，当混合浮动和流动布局时就容易遇到很多问题，下面结合示例介绍常见的问题和解决办法。

1．调整左右栏间距

【示例 1】制作左右两栏的页面，左栏是浮动布局，右栏是流动布局。代码如下，显示效果如图 9.20 所示。

```
<style type="text/css">
#contain {/*页面布局包含元素*/
    width:774px;                      /*定义页面宽*/
    border:double 4px #aaa;           /*定义页面边框*/
    padding:12px;                     /*为页面包含元素增加补白*/
    overflow:visible;                 /*定义包含元素自动伸缩显示所有包含内容*/
}
#contain img {/*定义左侧图片浮动显示*/
    width:200px;
    height:100px;
    float:left;
    clear:left;                       /*定义图片单列显示*/
    margin:0 12px 6px 0;              /*定义图片的边界*/
    padding:6px;
    border:solid 1px #999;
}
#contain h2 {     text-align:center;}     /*定义右侧标题居中*/
```



```
#contain p {/*定义段落属性*/
    margin:0;                      /*此时该属性左侧值最让人困惑，为什么？*/
    padding:0;                     /*此时该属性左侧值最让人困惑，为什么？*/
    line-height:1.8em;
    font-size:13px;
    text-indent:2em;
}
.clear {clear:both;}               /*定义清除类，处理非 IE 浏览器不能自适应包容问题*/
</style>
<div id="contain">
    <img src="bg10.jpg" /><img src="bg7.jpg" /><img src="bg2.jpg" /><img src="bg3.jpg" />
    <h2>《荷塘月色》（节选）</h2>
    <p>曲曲折折的荷塘上面，弥望的是田田的叶子。叶子出水很高，像亭亭的舞女的裙。…</p>
    <div class="clear"></div>
</div>
```



图 9.20　默认显示效果

上面的示例在图文混排的基础上利用浮动模型设计了一个更漂亮的图片通栏布局。使用 float 属性定义所有图片向左浮动，定义 clear 属性清除相邻图片并列浮动，将一组图片垂直排列。通过定义左栏浮动、右栏流动，保证页面中的文本围绕图片在右侧显示。

如果加大右侧文本与左侧图片之间的间距，一般用户会定义 p 元素的 margin-left 或 padding-left 属性值，但会发现为 p 元素定义左边界或左补白之后，左右栏间距没有变。

解决方法：不要定义流动元素的边界或补白，而是定义浮动元素的边界或补白，实现调控间距





的目的，因为浮动元素的边界和补白不会被流动元素覆盖。例如，定义浮动图像的右侧边界为 50 像素，代码如下，效果如图 9.21 所示。

```
margin:0 50px 6px 0;
```



图 9.21　显示效果

2. 调整上下栏间距

上下栏之间的浮动与流动混合布局也比较复杂，如上下栏间距不易调整、布局偶尔错乱等。

【示例 2】在示例 1 的基础上，给图文页面增加一个导航条。代码如下，显示结果如图 9.22 所示。

```
<style type="text/css">
body {/*定义窗口属性*/
    margin:0; /*清除 IE 默认边界属性值*/
    padding:0; /*清除非 IE 默认补白属性值*/
    }
#nav {/*定义导航列表框属性*/
    margin:0; /*清除 IE 默认缩进属性值*/
    padding:0; /*清除非 IE 默认缩进属性值*/
    list-style-type:none; /*清除浏览器默认列表样式*/
}
#nav li {/*定义菜单列表项显示效果*/
    float:left; /*向左浮动*/
    width:100px; height:32px;
    line-height:32px; /*垂直居中*/
```





```
    text-align:center; /*水平居中*/
    background:#7B9F23; /*背景色*/
    margin:1px; /*菜单间距*/
    font-size:14px;
}
#nav a {text-decoration:none;}/*定义导航链接属性*/
#contain {/*图文包含元素*/
    width:774px; /*定义图文框宽*/
    border:double 4px #aaa; /*定义图文框边框*/
    padding:12px; /*为图文框增加补白*/
    overflow:visible; /*定义图文框自动伸缩显示所有包含内容*/
}
#contain img {/*定义左侧图片浮动显示*/
    width:200px; height:100px;
    float:left; clear:left; /*定义图片单列显示*/
    margin:0 12px 6px 0; /*定义图片的边界*/
    padding:6px; border:solid 1px #999;
}
#contain h2 {text-align:center;}/*定义右侧标题居中*/
#contain p {/*定义段落属性*/
    margin:0; padding:0;
    line-height:1.8em; font-size:13px;
    text-indent:2em;
}
.clear {clear:both;} /*定义清除类，处理非 IE 浏览器不能自适应包容问题*/
</style>
<ul id="nav"><!--导航菜单模块-->
    <li><a href="">首页</a></li>
    <li><a href="">导航菜单</a></li>
    <li><a href="">导航菜单</a></li>
    <li><a href="">导航菜单</a></li>
    <li><a href="">导航菜单</a></li>
    <li><a href="">导航菜单</a></li>
    <li><a href="">导航菜单</a></li>
</ul>
<div id="contain"><!--图文框模块-->
    ……
</div>
```




Note



图 9.22　调整空隙

图 9.22 显示导航条已经进入下面的栏目内部，其解决方法是在列表项最后添加一个清除元素，强迫上面的 ul 元素自适应高度，以实现包含其内部的浮动列表项。代码如下。

```
<div class="clear"></div>
```

这样就不会出现浮动元素与流动包含元素相互脱节的现象，使浮动元素在上面栏目的包含框中，显示效果如图 9.23 所示。



图 9.23　正确显示效果







9.4 定位显示

定位布局的设计思路比较简单，它允许用户精确定义网页元素的显示位置，可以是绝对位置，也可以是相对位置。

9.4.1 定义定位显示

在 CSS 中，可以通过 position 属性定义元素定位显示，其语法如下。

```
position：static | relative | absolute | fixed
```

取值说明如下。

☑ static：表示不定位，元素遵循 HTML 默认的流动模型，如果未显式声明元素的定位类型，则默认为该值。

☑ relative：表示相对定位，它通过 left、right、top、bottom 属性确定元素在正常文档流中偏移位置。相对定位完成的过程是首先按 static 方式生成一个元素，然后移动这个元素，移动方向和幅度由 left、right、top、bottom 属性确定，元素的形状和偏移前的位置保留不动。

☑ absolute：表示绝对定位，将元素从文档流中拖出来，然后使用 left、right、top、bottom 属性相对于其最接近的一个具有定位属性的父定位包含框进行绝对定位。如果不存在这样的定位包含框，则相对于浏览器窗口，其层叠顺序则通过 z-index 属性来定义。

☑ fixed：表示固定定位，与 absolute 定位类型类似，但它的定位包含框是视图本身，由于视图本身是固定的，它不会随浏览器窗口的滚动条滚动而变化，除非在屏幕中移动浏览器窗口的位置或改变浏览器窗口的显示大小，因此固定定位的元素会始终位于浏览器窗口内视图的某个位置，不会受文档流动影响，这与 background-attachment:fixed;属性功能相同。

与浮动元素一样，绝对定位元素以块状显示，它会为所有子元素建立一个定位包含框，所有被包含元素都以定位包含框作为参照物进行定位或在其内部浮动和流动。

【示例 1】在本示例中，定义了 3 个不同模型的包含元素，然后观察不同模型的包含元素与它们的子元素的位置关系。代码如下，效果如图 9.24 所示。

```
<style type="text/css">
#contain1, #contain2, #contain3 {/*定义 3 个一级 div 元素对象的共同属性*/
    width: 380px;height: 120px; border: solid 1px #666;
    }
#contain2 {/*定义第 2 个一级 div 元素对象为绝对定位，并设置其距离窗口左边和上边的距离*/
    position: absolute; left: 120px; top: 60px; background: #F08080;
}
#contain3 {/*定义第 3 个一级 div 元素对象为浮动布局*/
    float: left; background: #D2B48C;
}
```




Note

```
#contain2 div {/*定义绝对定位对象内所有子元素对象的共同属性*/
    color: #993399; border: solid 1px #FF0000;
}
#sub_div1 {/*定义绝对定位对象包含框内第 1 个对象为绝对定位*/
    width: 80px; height: 80px; position: absolute;
    right: 10px; /*定义该绝对元素右边距离父级定位包含框的右边距离*/
    bottom: 10px; /*定义该绝对元素底边距离父级定位包含框的底边距离*/
    background: #FEF68F;
}
#sub_div2 {/*定义绝对定位对象包含框内第 2 个元素为浮动布局*/
    width: 80px; height: 80px; float: left; background: #DDA0DD;
}
#sub_div3 {/*定义绝对定位对象包含框内第 3 个元素的背景色、宽和高*/
    width: 100px; height: 90px; background: #CCFF66;
}
</style>
<div id="contain1">元素一流动</div>
<div id="contain2">元素二－绝对定位
    <div id="sub_div1">子元素 1－绝对定位</div>
    <div id="sub_div2">子元素 2－浮动</div>
    <div id="sub_div3">子元素 3－流动</div>
</div>
<div id="contain3">元素三－浮动</div>
```

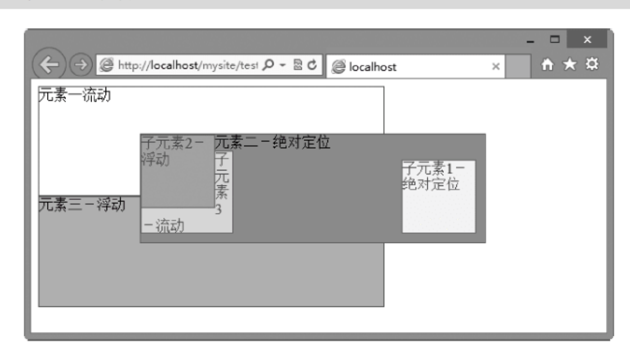


图 9.24　定位显示

从图 9.24 可以看到“元素二”被定义为绝对定位，它以浏览器窗口为定位包含框，显示位置根据元素左边到窗口左边的距离和元素上边到窗口上边的距离来确定。而“子元素 1”却以具有定位属性的“元素二”为定位包含框，显示位置根据“子元素 1”右边到“元素二”右边的距离和元素底边到“元素二”底边的距离来确定。在“元素二”中包含了 3 个子元素，它们以不同的性质显示，但它们都以“元素二”作为参照平台，包括其中的浮动元素和绝对定位元素。





如果把行内元素作为定位包含框，情况就很复杂，因为行内元素有可能在几行内显示，产生好几个线性盒，这时定位包含框就被定义为这几行区域，而其内部被包含的绝对定位子元素将根据行内元素的第 1 行第 1 个字符左上角来确定 left 和 top 属性的偏移值，根据第一行最后一个字符的右下角来确定 right 和 bottom 属性的偏移值。

【示例 2】在本示例中，在文本段中设置两个相互嵌套的 span 元素，然后把外层的 span 元素定义为定位包含框，而把内层的 span 元素定义为绝对定位，并进行偏移定位。代码如下，显示效果如图 9.25 所示。

```
<style type="text/css">
p {/*定义文本段属性*/
    width: 400px; height: 200px;
    border: dashed 1px green;
}
#relative {/*定义定位包含框，并用蓝色线框标示*/
    position: relative;
    border: solid 1px blue;
}
#absolute {/*定义绝对定位子元素，并向右下角偏移 200px*/
    position: absolute;
    left: 200px; top: 200px;
    border: solid 2px red;
}
</style>
<p> 在用 CSS 控制排版过程中，<span id="relative">定位一直被人认为是一个难点，这主要是表现为很多网友在没有深入理解清楚定位的原理时，排出来的杂乱网页常让他们不知所措，而另一边一些高手则常常借助定位的强大功能做出些很酷的效果来，<span  id="absolute">比如 CSS 相册等等</span>，因此自己杂乱的网页与高手完美的设计形成鲜明对比，</span>这在一定程度上打击了初学定位的网友，也在他们心目中形成这样的一种思想：当我熟练地玩转 CSS 定位时，我就已是高手了。 </p>
```

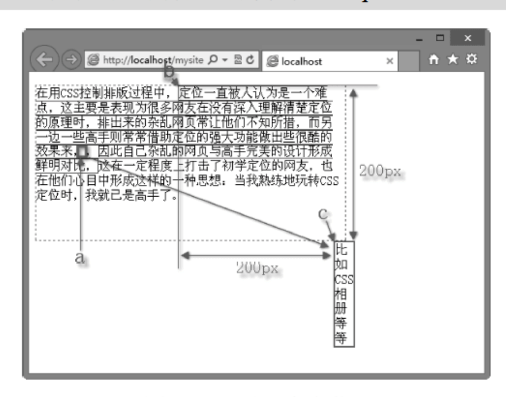


图 9.25　子元素定位 1





在图 9.25 中，a 所指示的位置为子元素定位前的位置，b 所指示的顶角为子元素偏移的参考点，c 所指示的顶角为偏移后的定位点。可以看到，在定位包含框多行显示时，内部定位元素将根据 b 点（第 1 行第 1 个字的左上角）进行偏移定位，而不是以文本段左上角为参考点进行定位。

如果定义 right 和 bottom 属性进行偏移（其中 CSS 代码如下），内部定位元素将根据 b 点（第 1 行最后 1 个字的右下角）进行偏移定位，而不是以文本段右下角为参考点进行定位。

```
#absolute {/*定义绝对定位子元素，并向左上角偏移 100px*/
    position:absolute;
    right:100px;
    bottom:100px;
    border:solid 2px red;
}
```



9.4.2 定位框

CSS 定位包含框是标准布局中的一个重要概念，它是绝对定位的基础。注意，要区分定位包含框与父元素、包含框或包含元素等概念。

定位包含框就是为绝对定位元素提供坐标偏移和显示范围的参照物，即确定绝对定位的偏移起点和百分比长度的参考。在默认状态下，body 元素就是一个根定位包含框，所有绝对定位的元素就是根据窗口来确定自己所处的位置和百分比大小显示的。但是如果定义了包含元素为定位包含框以后，对于被包含的绝对定位元素来说，就会根据最接近的具有定位功能的上级包含元素来决定自己的显示位置。

【示例】为了能直观地理解定位包含框，本示例先构建一个 HTML 代码模块。

```
<div id="a">
    <div id="c"></div>
</div>
<div id="b">
    <div id="d"></div>
</div>
```

在上面的代码中，构建了两个定位包含框，它们分别包含了一个元素。下面用 CSS 定义这两个包含元素的大小为 200px *200px，并浮动在窗口的中间区域，代码如下。

```
#a,#b {/* 定义包含元素的共同属性 */
    width:200px;
    height:200px;
    float:left;
    margin-top:50px;            /* 拉开与窗口顶部的距离 */
    border:solid 1px red;       /* 定义红色边框线，便于识别 */
}
```





同时单独定义 b 包含元素为相对定位，确定它是一个定位包含框，代码如下。

```
#b {/* 定义包含元素 b 为相对定位，确定它为定位包含框 */
    position:relative;
    margin-left:50px;                /* 拉开与 b 包含元素的距离 */
}
```

然后，定义两个被包含元素为绝对定位，大小为 50%*50%，并都偏移 50%，代码如下。

```
#c,#d {/* 定义被包含元素绝对定位，并进行偏移 */
    width:50%;
    height:50%;
    position:absolute;
    left:50%;                        /* 与定位包含框左侧边框距离为 50% */
    top:50%;                         /* 与定位包含框顶部边框距离为 50% */
}
```

最后，分别为两个被包含元素定义不同的背景颜色，以便于区别，显示效果如图 9.26 所示。

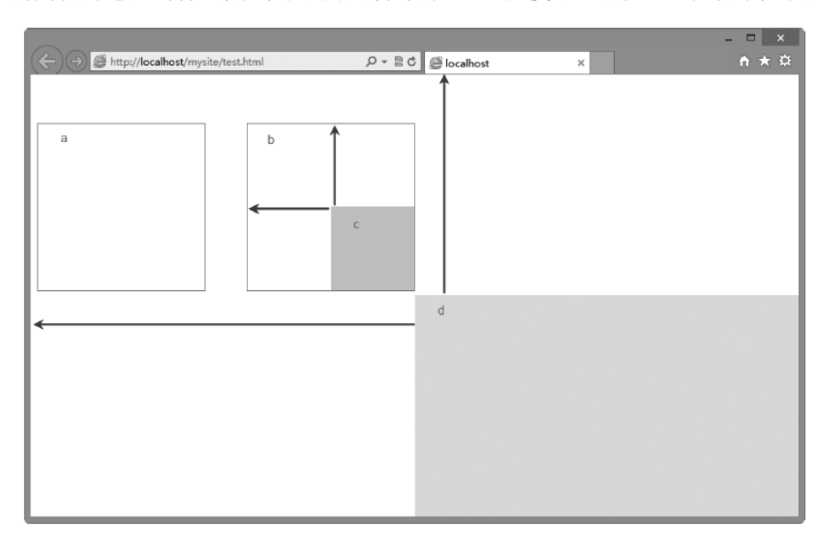


图 9.26　定义定位包含框

在图 9.26 所示的演示效果中，被 a 包含元素包含的 c 子元素以窗口 body 元素的左上点为坐标原点进行绝对定位偏移，百分比大小取值也根据窗口的大小来确定，即为窗口宽度和高度的一半。

而 b 包含元素被定义为相对定位，它就成为一个定位包含框，因此，被它包含的 d 元素就会以 b 元素的左上角为坐标原点进行绝对定位偏移，它的百分比大小取值也会根据 b 元素的大小来确定，而不是以窗口为参照物。

但是，在 IE 早期版本浏览器中对于被包含的绝对定位元素的百分比大小解析依然存在问题，如图 9.27 所示。





Note

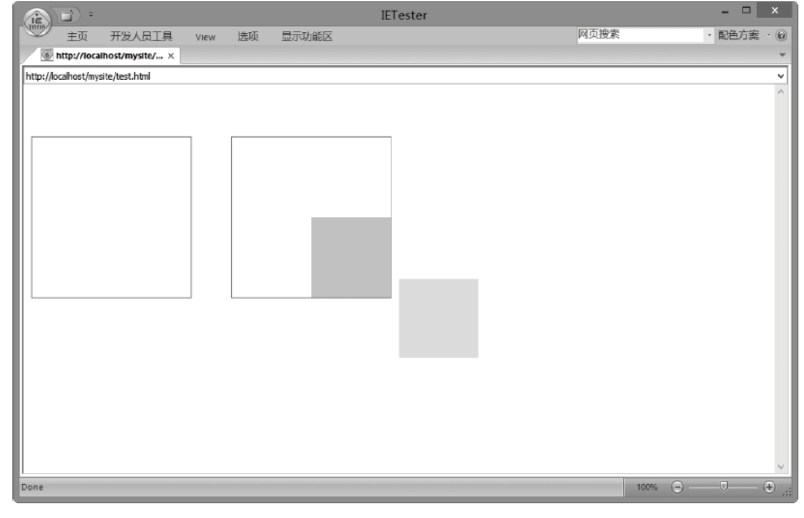


图 9.27　IE 中的定位包含框显示效果

图 9.27 是上面示例在 IE 6 浏览器中的预览效果，可以看到，对于坐标偏移解析方面，IE 与其他现代标准浏览器的解析效果是一致的，即 a 包含元素内的 c 元素根据最近定位包含框（窗口左上角）进行偏移，百分比偏移大小（50%）也是根据定位包含框大小（窗口大小）来确定的。但是，在计算被包含元素自身大小时，IE 6 与标准存在很大的差异，IE 6 浏览器认为被包含元素 c 的百分比高和宽应该根据 HTML 代码中包含它的元素的大小来确定，而不是根据它的最近定位包含框来确定，因此 c 元素显示大小（100px*100px）就为 a 元素显示大小（200px*200px）的一半。

一般情况下，可以用 position 属性来定义任意定位包含框，position 属性有效取值包括 absolute、fixed 和 relative。

有了定位包含框，就可以灵活设置绝对定位的坐标原点和它的参考值。绝对定位打破了元素的固有排列顺序，满足诸如内容优先的排版需要，也给复杂的浮动布局带来方便。



视 频 讲 解

9.4.3　相对定位

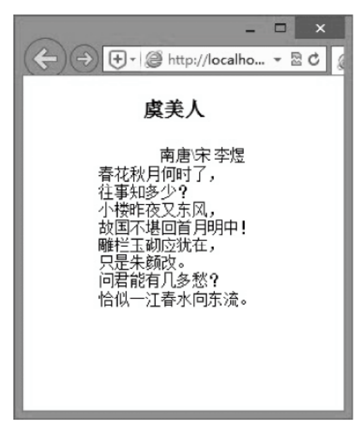
与绝对定位不同的是，相对定位元素的偏移量是根据它在正常文档流里的原始位置计算的，而绝对定位元素的偏移量是根据定位包含框的位置计算的。一个绝对定位元素的位置取决于它的偏移量 top、right、bottom 和 left 属性值，相对定位元素的偏移量与绝对定位一样。

【示例】在本示例中，定义 strong 元素对象为相对定位，然后通过相对定位调整标题在文档顶部的显示。代码如下，显示效果如图 9.28 所示。

```
<style type="text/css">
p { margin: 60px; font-size: 14px;}
p span { position: relative; }
p strong {/*[相对定位]*/
    position: relative;
    left: 40px; top: -40px;
```



```
        font-size: 18px;
    }
    </style>
    <p> <span><strong>虞美人</strong>南唐\宋 李煜</span> <br>春花秋月何时了，<br>往事知多少？<br>小楼昨夜又东风，<br>故国不堪回首月明中！<br>雕栏玉砌应犹在，<br>只是朱颜改。<br>问君能有几多愁？<br>恰似一江春水向东流。 </p>
```





定位前　　　　定位后

图 9.28　相对定位显示效果

从图 9.28 可以看到，相对定位后，元素对象的原空间保留不变。相对定位偏离的边距遵循绝对定位中的偏离规则，不过相对定位的定位包含框是元素对象的原位置。



提示： 相对定位元素遵循的是流动布局模型，存在于正常的文档流中，但是它的位置可以根据原位置进行偏移。由于相对定位元素占有自己的空间，即原始位置保留不变，因此它不会挤占其他元素的位置，但可以覆盖在其他元素之上进行显示。

与相对定位元素不同，绝对定位元素完全被拖离正常文档流中原来的空间，且原来空间将不再保留，而是被相邻元素挤占。把绝对定位元素设置在可视区域之外会导致浏览器窗口的滚动条出现。而设置相对定位元素在可视区域之外，滚动条是不会出现的。



9.4.4　定位层叠

定位元素之间可以重叠显示，这与图像合成类似，而在流动布局和浮动布局中是无法实现这种重叠效果的，因此利用定位重叠技术可以创建动态网页效果。

在 CSS 中可以通过 z-index 属性来确定定位元素的层叠等级。需要声明的是，z-index 属性只有在元素的 position 属性取值为 relative、absolute 或 fixed 时才可以使用。其中 fixed 属性值目前还没有得到 IE 的支持。

【示例 1】 在本示例中，定义两个定位元素，然后通过 z-index 属性调整它们的层叠顺序。代码如下，效果如图 9.29 所示。




Note

```
<style type="text/css">
#sub_1,#sub_2 {/*定义子元素绝对定位，并设置宽和高*/
 position: absolute;
 width:200px; height:200px;
}
#sub_1 {/*定义第 1 个子元素的属性*/
 z-index:10; /*设置层叠等级为 10*/
 left:50px; top:50px;
 background:red;
}
#sub_2 {/*定义第 2 个子元素的属性*/
 z-index:1; /*设置层叠等级为 1*/
 left:20px; top:20px;
 background:blue;
}
</style>
<div id="contain">
   <div id="sub_1">元素 1</div>
   <div id="sub_2">元素 2</div>
</div>
```

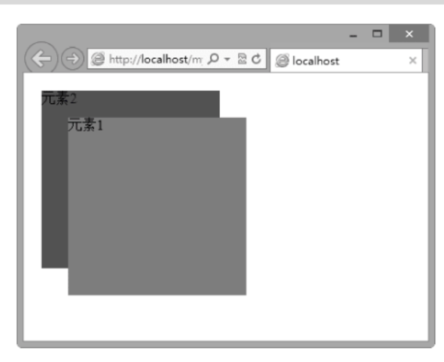


图 9.29　层叠定位显示

z-index 属性值越大，层叠级别就越高，如果属性值相同，则根据结构顺序来计算层叠级别。对于未指定此属性的绝对定位元素，此属性的 number 值为正数的元素会在其之上，而 number 值为负数的元素在其之下。此属性仅仅作用于 position 属性值为 relative、absolute 或 fixed 的元素。

【示例 2】如果 z-index 属性值为负值，则将隐藏在文档流的下面。在本示例中，定义<div>标签的相对定位，并设置 z-index 属性值为-1，显示效果如图 9.30 所示。

```
<!doctype html>
<html>
```





```
<head>
<meta charset="utf-8">
<style type="text/css">
#box1 {
    height: 400px;                       /* 固定高度 */
    position: relative;                  /* 相对定位 */
    background: red url(images/1.jpg);   /* 定义背景色和背景图 */
    z-index: -1;                         /* 层叠顺序*/
    top: -120px;                         /* 偏移位置，实现与文本重叠显示 */
}
</style>
</head>
<body>
<p>我永远相信只要永不放弃，我们还是有机会的。最后，我们还是坚信一点，这世界上只要有梦想，只要不断努力，只要不断学习，不管你长得如何，不管是这样，还是那样，男人的长相往往和他的才华成反比。今天很残酷，明天更残酷，后天很美好，但绝大部分是死在明天晚上，所以每个人都不要放弃今天。</p>
<div id="box1"></div>
</body>
</html>
```



图 9.30　定义定位元素显示在文档流下面

9.4.5　混合定位布局



混合定位是利用相对定位的流动模型优势和绝对定位的层布局优势，实现网页定位的灵活性和精确性优势互补。例如，如果给父元素定义为 position:relative，给子元素定义为 position:absolute，那么子元素的位置将随着父元素，而不是随着整个页面进行变化。

【示例】本示例利用混合定位布局方法，设计了一个 3 行 2 列的页面。代码如下，效果如图 9.31 所示。




```
<style type="text/css">
body {/*定义窗体属性*/
    margin: 0;                  /*清除 IE 默认边距*/
    padding: 0;                 /*清除非 IE 默认边距*/
    text-align: center;         /*设置在 IE 浏览器中居中对齐*/
}
#contain {/*定义父元素为相对定位，实现定位包含框*/
    width: 100%;                /*定义宽度*/
    height: 310px;              /*必须定义父元素的高度，该高度应大于绝对布局的最大高度，否则父元
                                素背景色就无法显示，且后面的布局区域也会无法正确显示*/
    position: relative;         /*定义为相对定位*/
    background: #E0EEEE;
    margin: 0 auto;             /*非 IE 浏览器中居中显示*/
}
#header, #footer {/*定义头部和脚部区域属性，以默认的流动模型布局*/
    width: 100%;
    height: 50px;
    background: #C0FE3E;
    margin: 0 auto;             /*非 IE 浏览器中居中显示*/
}
#sub_contain1 {   /*定义左侧子元素为绝对定位*/
    width: 30%;                 /*根据定位包含框定义左侧栏目的宽度*/
    position: absolute;         /*定义子栏目为绝对定位*/
    top: 0;                     /*在定位包含框顶边对齐*/
    left: 0;                    /*在定位包含框左边对齐*/
    height: 300px;              /*定义高度*/
    background: #E066FE;
}
#sub_contain2 { /*定义右侧子元素为绝对定位*/
    width: 70%;                 /*根据定位包含框定义右侧栏目的宽度*/
    position: absolute;         /*定义子栏目为绝对定位*/
    top: 0;                     /*在定位包含框顶边对齐*/
    right: 0;                   /*在定位包含框右边对齐*/
    height: 200px;              /*定义高度*/
    background: #CDCD00;
}
</style>
<div id="header">标题栏</div>
```





```
<div id="contain">
    <div id="sub_contain1">左栏</div>
    <div id="sub_contain2">右栏</div>
</div>
<div id="footer">页脚</div>
```



图 9.31　混合定位演示效果

在上面示例中，设计左右栏为绝对定位显示，两栏包含框为相对定位显示，这样左右栏就以包含框为定位参考。由于定位包含框的高度不会随子元素的高度而变化，因此要实现合理布局，必须给父元素定义一个明确的高度才能显示包含框背景，后面的布局元素也才能跟随绝对定位元素之后正常显示。



9.5　案例实战

本节将通过多个案例帮助用户快速掌握网页布局的基本方法和思路，为网页实战奠定扎实的基础。所有案例都包含 5 个模块的典型标准结构，下面借助这个结构来尝试设计不同的版式效果。通过这种方法帮助用户快速掌握多种布局效果的设计思路和实现方法，代码如下。

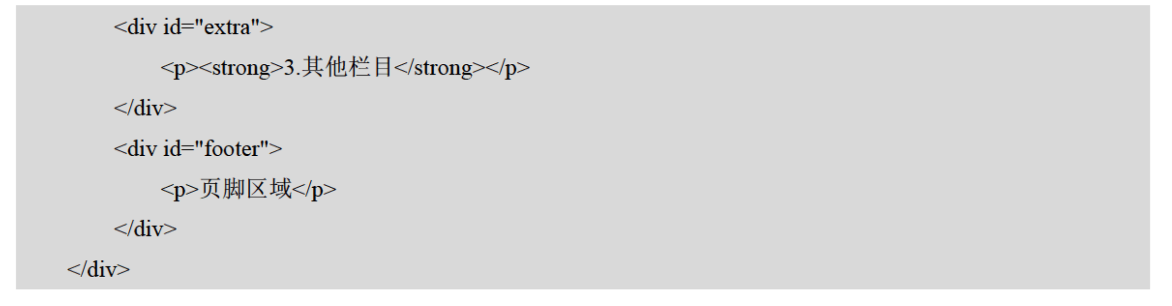
```
<div id="container">
    <div id="header">
        <h1>页眉区域</h1>
    </div>
    <div id="wrapper">
        <div id="content">
            <p><strong>1.主体内容区域</strong></p>
        </div>
    </div>
    <div id="navigation">
        <p><strong>2.导航栏</strong></p>
    </div>
```





Note

视频讲解



```
        <div id="extra">
            <p><strong>3.其他栏目</strong></p>
        </div>
        <div id="footer">
            <p>页脚区域</p>
        </div>
    </div>
```

9.5.1 设计固宽+弹性页面

本案例版式设置的导航栏与其他栏并为一列固定在右侧，主栏目以弹性方式显示在左侧，实现主栏自适应页面宽度变化，而侧栏宽度固定不变的版式效果，如图 9.32 所示。

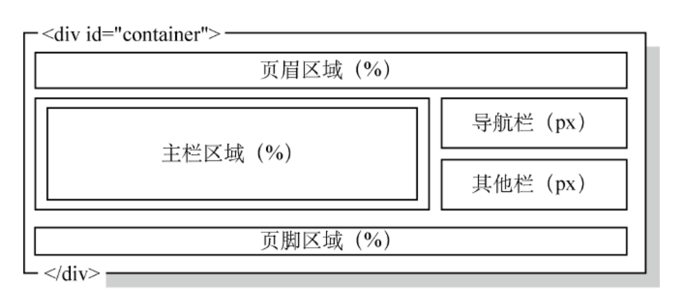


图 9.32　版式结构示意图

如果完全使用浮动布局来设计主栏自适应、侧栏固定的版式是存在很大难度的，因为百分比取值是一个不固定的宽度，让一个不固定宽度的栏目与一个固定宽度的栏目同时浮动在一行内，采用简单的方法是不行的。

这里设计主栏为 100%宽度，然后通过左外边距取负值强迫栏目偏移出一列的空间，最后把这个腾出的区域让给右侧浮动的侧栏，从而达到并列浮动显示的目的。

当主栏左外边距取负值时，可能部分栏目内容显示在窗口外面，为此在嵌套的子元素中设置左外边距为父包含框的左外边距的负值，这样就可以把主栏内容控制在浏览器的显示区域了。

本示例的样式代码如下，设计效果如图 9.33 所示。

```
div#wrapper {/* 主栏外框 */
    float:left;                          /* 向左浮动 */
    width:100%;                          /* 弹性宽度 */
    margin-left:-200px                   /* 左侧外边距，负值向左缩进 */
}
div#content {/* 主栏内框 */
    margin-left:200px                    /* 左侧外边距，正值填充缩进 */
}
div#navigation {/* 导航栏 */
```





```
        float:right;                          /* 向右浮动 */
        width:200px                           /* 固定宽度 */
    }
    div#extra {/* 其他栏 */
        float:right;                          /* 向右浮动 */
        clear:right;                          /* 清除右侧浮动，避免同行显示 */
        width:200px                           /* 固定宽度 */
    }
    div#footer {/* 页眉区域 */
        clear:both;                           /* 清除两侧浮动，强迫外框撑起 */
        width:100%                            /* 宽度 */
    }
```

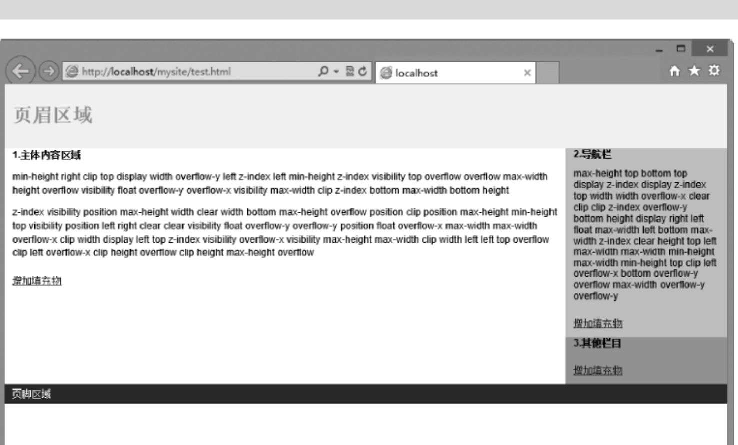


图 9.33　设计固宽+自适应两栏页面

9.5.2　设计两栏弹性页面



在两栏浮动版式中，如果设置两列宽度都为自适应，那么设置起来会容易得多。例如，定义两栏版式中主栏向左浮动，宽度为 70%，导航栏向右浮动，宽度为 29.9%，代码如下。

```
    div#wrapper {
        float:left;                           /* 向左浮动 */
        width:70%                             /* 百分比宽度 */
    }
    div#navigation {
        float:right;                          /* 向右浮动 */
        width:29.9%                           /* 百分比宽度 */
    }
    div#extra {
```




```
    clear:both;                             /* 清除左右浮动 */
    width:100%                              /* 满屏显示 */
}
```



下面的示例将设计一个更精确的两栏浮动且自适应宽度的版式。

本案例版式设置的导航栏与其他栏目并为一列固定在右侧，主栏目以弹性方式显示在左侧，实现主栏自适应页面宽度变化，而侧栏宽度固定不变的版式效果，如图 9.34 所示。

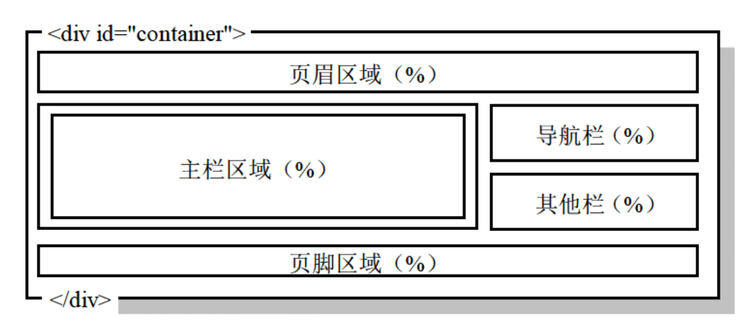


图 9.34　版式结构示意图

设计的方法也是采用负外边距来进行调节，核心样式如下，详细代码请参阅本书实例。

```
div#wrapper {/* 主栏外框 */
    float:right;                            /* 向右浮动 */
    width:100%;                             /* 弹性宽度 */
    margin-right:-33%;                      /* 右侧外边距，负值向右缩进 */
}
div#content {/* 主栏内框 */
    margin-right:33%;                       /* 右侧外边距，正值填充缩进 */
}
div#navigation {/* 导航栏 */
    float:left;                             /* 向右浮动 */
    width:32.9%;                            /* 固定宽度 */
}
div#extra {/* 其他栏 */
    float:left;                             /* 向左浮动 */
    clear:left;                             /* 清除左侧浮动，避免同行显示 */
    width:32.9%                             /* 固定宽度 */
}
div#footer {/* 页眉区域 */
    clear:both;                             /* 清除两侧浮动，强迫外框撑起 */
    width:100%                              /* 宽度 */
}
```





为了避免在 IE 7 或者其他标准浏览器窗口中出现 y 轴滚动条，可以为 body 元素增加 overflow-x:hidden;声明隐藏该滚动条。代码如下，最后所设计的效果如图 9.35 所示。

```
body { overflow-x:hidden;}
```

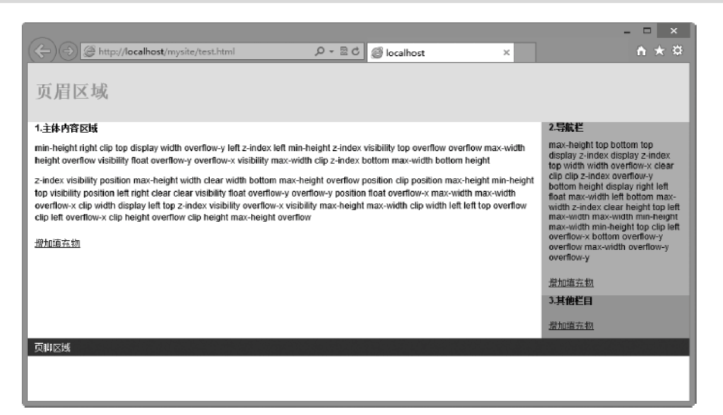


图 9.35　设计两栏宽度自适应页面

9.5.3　设计两栏浮动页面



上面两节分别采用不同的设计思路来设计两栏浮动版式，下面介绍如何使用另一种思维来设计两栏浮动版式。整个版式设置导航栏固定在左侧，主栏目以弹性方式显示在右侧，以实现主栏自适应页面宽度变化，而其他栏目与页脚区域显示底部的版式效果，如图 9.36 所示。

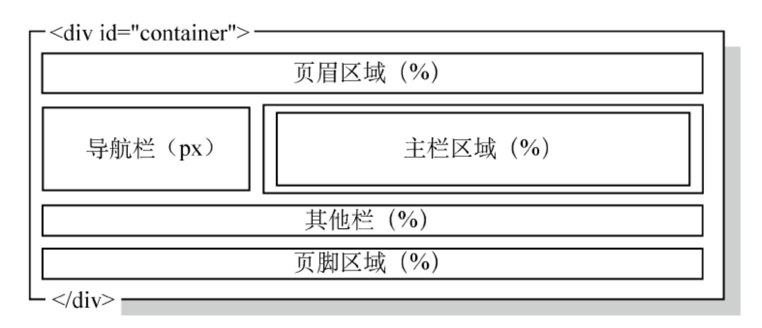


图 9.36　版式结构示意图

设计的方法是：让主栏（<div id="wrapper">）满屏显示，即取值为 100%，然后设置其包含的子元素（<div id="content">）左侧外边距为 200 像素，预留出一块区域。最后定义导航栏（<div id="navigation">)取值为−100%，也就是强制其从右侧的窗口外边移动到主栏左侧的预留区域内显示。所设计版式的核心代码如下，预览效果如图 9.37 所示。

```
div#wrapper {/* 主栏外包含框 */
    float:left;                          /* 向左浮动 */
    width:100%                           /* 满屏宽度 */
}
```

Note

```
div#content {/* 主栏内包含框 */
    margin-left:200px                          /* 左侧外边距，预留空间 */
}
div#navigation {/* 导航栏 */
    float:left;                                /* 向左浮动 */
    width:200px; /* 固定宽度，保持与主栏左侧预留区域的宽度一致 */
    margin-left:-100% /* 通过取左外边距负值，向左强制移动到主栏左侧的预留区域 */
}
div#extra {/* 其他栏 */
    clear:left;                                /* 清除左右浮动 */
    width:100%                                 /* 固定宽度 */
}
```

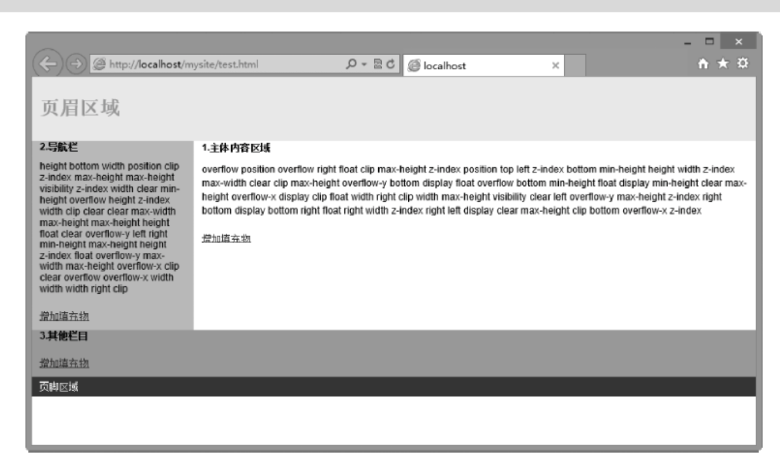


图 9.37　设计双浮动兼容页面



视频讲解

9.5.4　设计三栏弹性页面

本案例通过浮动布局的方法，以百分比为单位来设置栏目的宽度，版式结构示意图如图 9.38 所示。

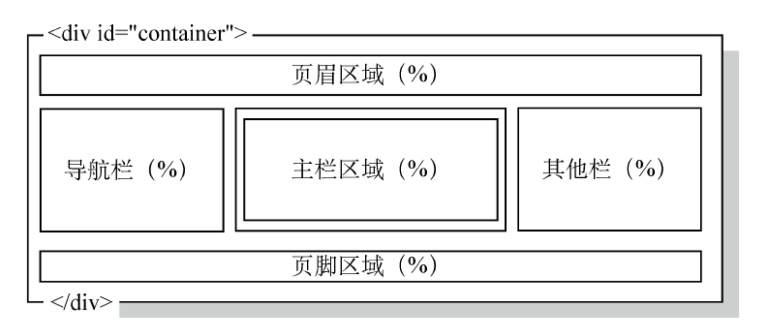


图 9.38　三列弹性版式结构示意图





本案例采用负外边距的方法来进行设计，这里设计三列都向左浮动，然后通过负外边距来定位每列的显示位置，布局示意图如图 9.39 所示。

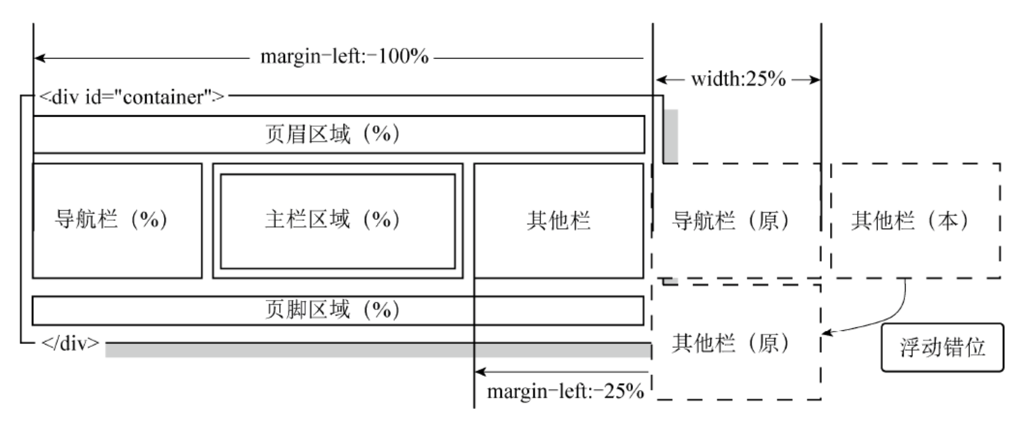


图 9.39　三列弹性版式布局示意图

注意，由于其他栏目在不受外界干扰的情况下会浮动在导航栏的右侧，但是并列浮动的总宽度超出了窗口宽度，因此会发生错位现象。而在没有负外边距的影响下，则会显示在第 2 行的位置，通过外边距取负值，强迫它们显示在主栏区域的上面。核心样式如下。

```
div#wrapper { /* 主栏外包含框基本样式 */
    float:left;                          /* 向左浮动 */
    width:100%                           /* 百分比宽度 */
}
div#content {/* 主栏内包含框基本样式 */
    margin: 0 25%                        /* 在左右两侧预留侧栏空间 */
}
div#navigation {/* 导航栏基本样式 */
    float:left;                          /* 向左浮动 */
    width:25%;                           /* 百分比宽度 */
    margin-left:-100%                    /* 左外边距取负值进行定位 */
}
div#extra {/* 其他栏基本样式 */
    float:left;                          /* 向左浮动 */
    width:25%;                           /* 百分比宽度 */
    margin-left:-25%                     /* 左外边距取负值进行定位 */
}
div#footer {/* 页脚包含框样式 */
    clear:left;                          /* 清除左右浮动 */
    width:100%                           /* 百分比宽度 */
}
```







三列弹性布局的版式设计效果如图 9.40 所示。

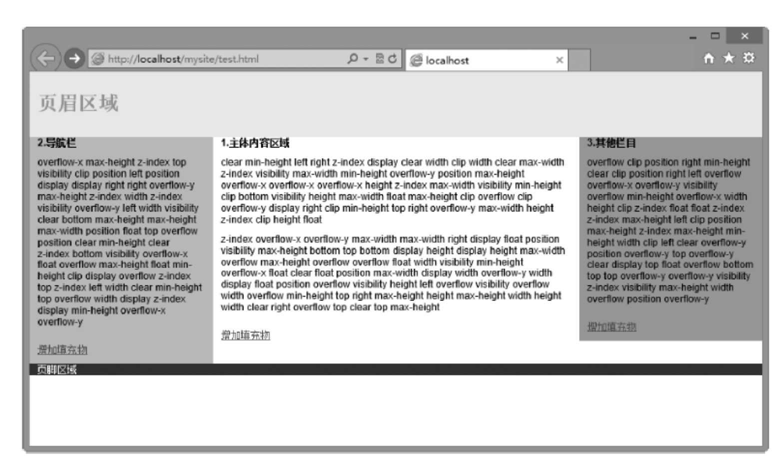
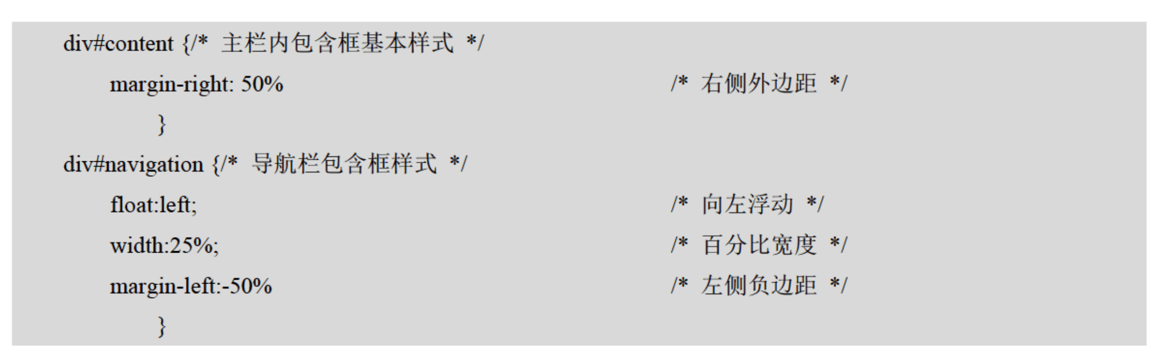


图 9.40　三列弹性版式的布局效果（一）

以同样的设计方法，如果设置侧栏负边距为其他值，则可以设置不同的版式效果。例如，在上面示例的基础上，设置主栏右侧外边距为 50%，定义导航栏左外边距负值为−50%，代码如下，则会显示如图 9.41 所示的效果。



```
div#content {/* 主栏内包含框基本样式 */
    margin-right: 50%                          /* 右侧外边距 */
        }
div#navigation {/* 导航栏包含框样式 */
    float:left;                                /* 向左浮动 */
    width:25%;                                 /* 百分比宽度 */
    margin-left:-50%                           /* 左侧负边距 */
        }
```

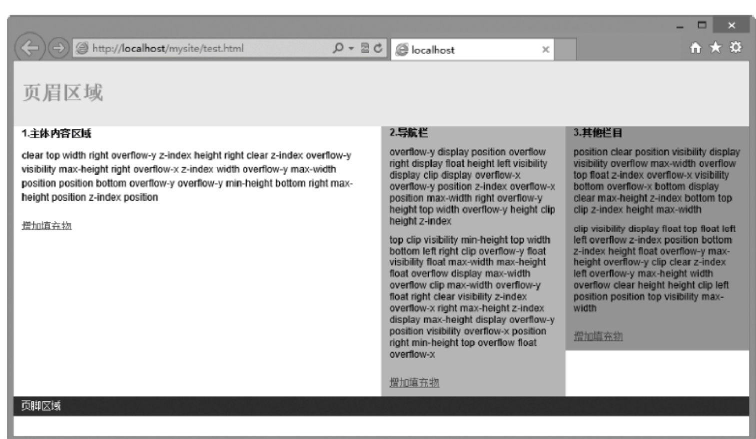


图 9.41　三列弹性版式的布局效果（二）



同样的道理，如果稍稍改变这几个包含框的外边距，会发现网页版式又发生了新的变化。例如，把主栏包含框的左外边距设置为 50%，通过负外边距让导航栏包含框向左移动 75%的距离，而让其他栏目移动 100%的距离，代码如下，则会显示如图 9.42 所示的效果。

```
div#content {/* 主栏内包含框的基本样式 */
    margin-left: 50%                    /* 左侧外边距 */
        }
div#navigation {/* 导航栏包含框的基本样式 */
    float:left;                         /* 向左浮动 */
    width:25%;                          /* 百分比宽度 */
    margin-left:-75%                    /* 左侧负边距 */
        }
div#extra {/* 其他栏包含框的基本样式 */
    float:left;                         /* 向左浮动 */
    width:25%;                          /* 百分比宽度 */
    margin-left:-100%                   /* 左侧负边距 */
        }
```

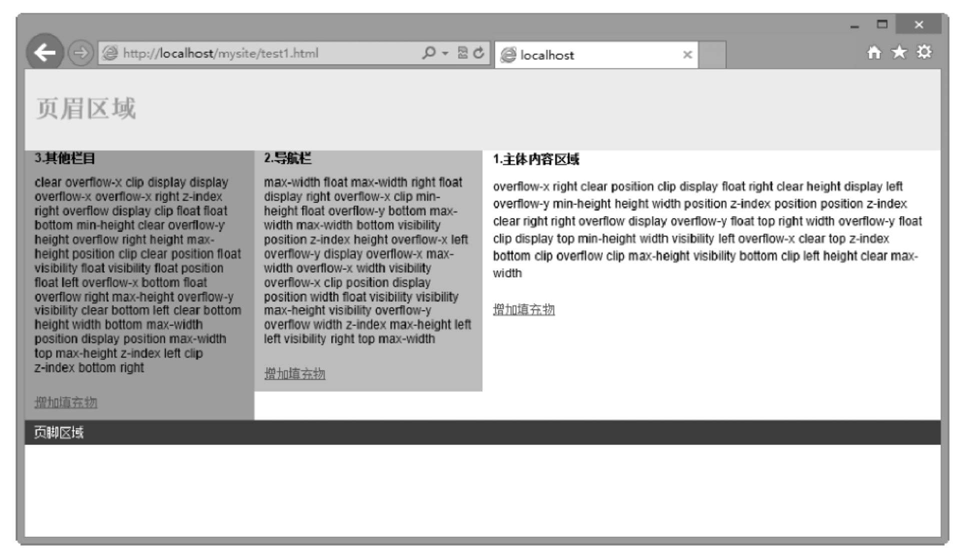


图 9.42　三列弹性版式的布局效果（三）

9.5.5　设计两列固宽+单列弹性页面



单纯的弹性或者固定版式布局相对来说都比较好控制，但是如果要设计一列弹性、另两列固定的版式就比较麻烦。不过灵活使用负外边距在网页布局中的技巧，可以解决类似复杂的布局。

本案例网页继续沿用上一节的模板示例结构，通过浮动布局的方法，以百分比和像素为单位来设置栏目的宽度，版式结构示意图如图 9.43 所示。





Note

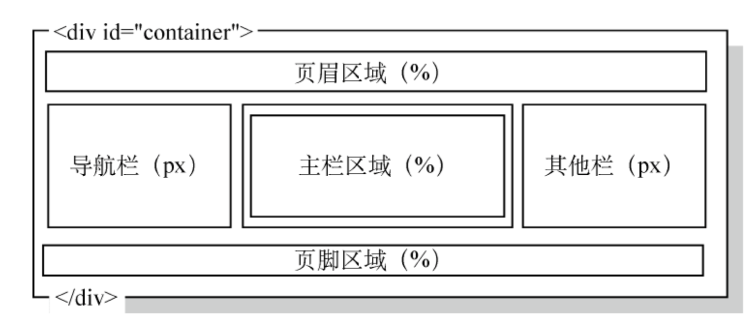


图 9.43　一列弹性两列固定版式结构示意图

要定义导航栏和其他栏宽度固定，不妨选用像素为单位，对于主栏则可以采用百分比单位，然后通过负外边距来定位每列的显示位置。布局示意图如图 9.44 所示。

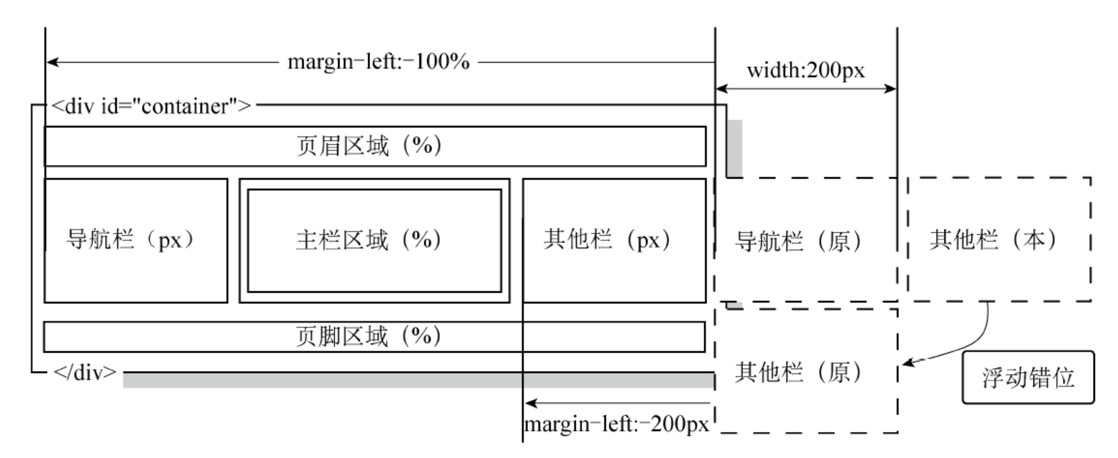


图 9.44　一列弹性两列固定版式布局示意图

注意，由于其他栏目在不受外界干扰的情况下会浮动在导航栏的右侧，但是如果并列浮动的总宽度超出了窗口宽度，就会发生错位现象。而在没有负外边距的影响下，则会显示在第 2 行的位置，通过外边距取负值，强迫它们显示在主栏区域的上面。核心样式如下。

```
div#wrapper { /* 主栏外包含框基本样式 */
    float:left;                          /* 向左浮动 */
    width:100%                           /* 百分比宽度 */
}
div#content {/* 主栏内包含框基本样式 */
    margin: 0 200px                      /* 在左右两侧预留侧栏空间 */
}
div#navigation {/* 导航栏基本样式 */
    float:left;                          /* 向左浮动 */
    width:200px;                         /* 固定宽度 */
    margin-left:-100%                    /* 左外边距取负值进行定位 */
```





```
}
div#extra {/*  其他栏基本样式  */
    float:left;                          /*  向左浮动  */
    width:200px;                         /*  固定宽度  */
    margin-left:-200px                   /*  左外边距取负值进行定位  */
}
```

一列弹性两列固定版式的布局效果如图 9.45 所示。

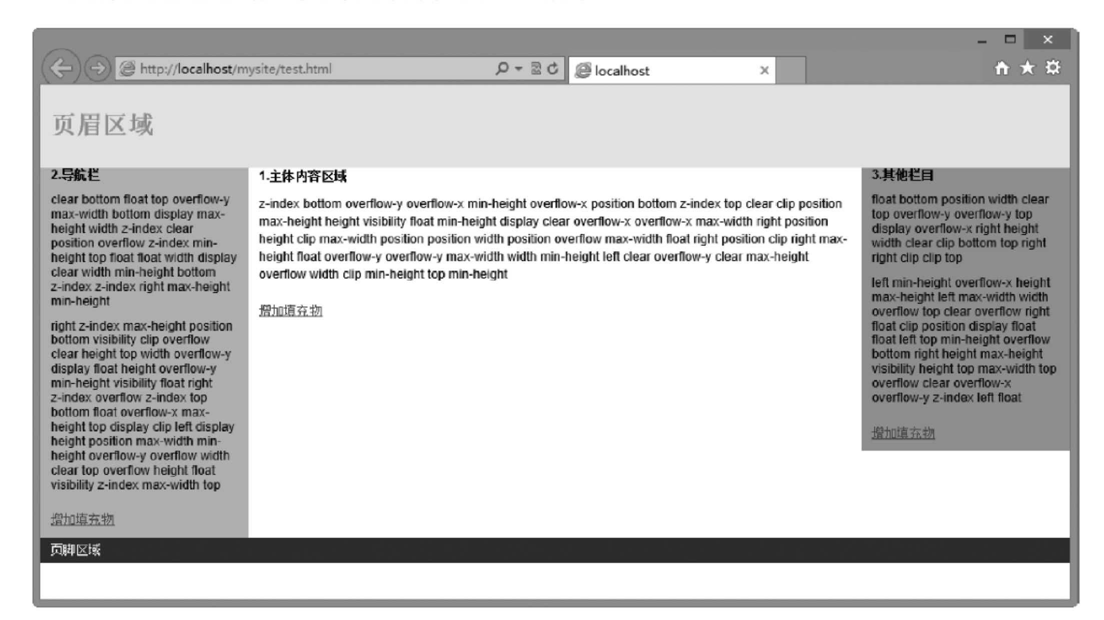


图 9.45　设计一列弹性两列固定版式的布局效果（一）

利用上面的设计技巧，也可以设计很多类似的版式效果。例如，分别调整侧栏和主栏的外边距取值，代码如下，效果如图 9.46 所示。

```
div#content { /*  主栏外包含框基本样式  */
    margin-right: 400px                  /*  通过左右外边距预留侧栏空间  */
}
div#navigation {/*  导航栏基本样式  */
    margin-left:-200px                   /*左外边距取负值进行精确定位*/
}
div#extra {/*  其他栏基本样式  */
    margin-left:-400px                   /*左外边距取负值进行精确定位*/
}
```





Note

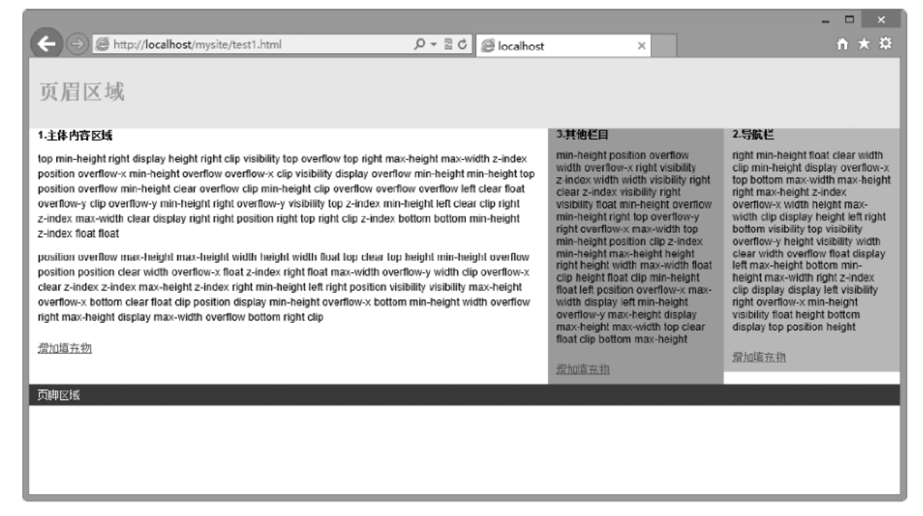


图 9.46　设计一列弹性两列固定版式的布局效果（二）



视频讲解

9.5.6　设计两列弹性+单列固定页面

与一列弹性两列固定版式相比，两列弹性一列固定版式似乎显得多余，不过当设计一个双主题的页面或者两列栏目都很重要的页面时，使用两列弹性一列固定版式进行布局会让页面更具灵活性。在设计思路上两列弹性一列固定版式与一列弹性两列固定版式大同小异，相信通过这样单独的分解，更能够引起您的重视和思考。

本案例的基本思路：首先定义主栏外包含框宽度为100%，即占据整个窗口。然后通过左右外边距来定义两侧空白区域，预留给侧栏占用。在设计外边距时，一侧采用百分比单位，另一侧采用像素为单位，这样就可以设计出两列宽度是弹性的、另一列是固定的。最后通过负外边距来定位侧栏的显示位置。代码如下，设计效果如图 9.47 所示。

```
div#wrapper {/* 主栏外包含框基本样式 */
    float:left;                          /* 向左浮动 */
    width:100%                           /* 百分比宽度 */
}
div#content {/* 主栏内包含框基本样式 */
    margin: 0 33% 0 200px                /* 定义左右两侧外边距，注意不同的取值单位 */
}
div#navigation {/* 导航栏包含框基本样式 */
    float:left;                          /* 向左浮动 */
    width:200px;                         /* 固定宽度 */
    margin-left:-100%                    /*左外边距取负值进行精确定位*/
}
div#extra {/* 其他栏包含框基本样式 */
    float:left;                          /* 向左浮动 */
```





```
        width:33%;                          /* 百分比宽度 */
        margin-left:-33%                    /*左外边距取负值进行精确定位*/
    }
```

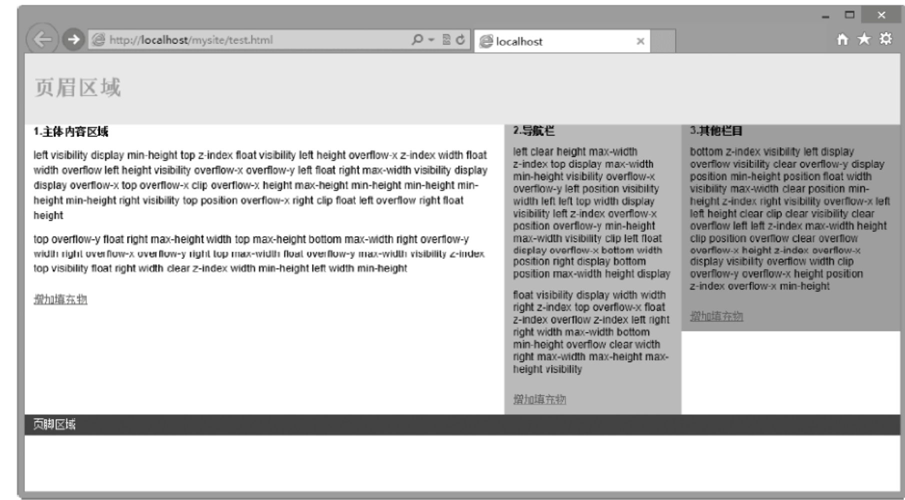


图 9.47　设计两列弹性一列固定版式的布局效果（一）

也可以让主栏取负外边距进行定位，其他栏自然浮动。例如，修改其中的核心代码，让主栏外包含框向左取负值，偏移 25%的宽度，也就是隐藏主栏外框左侧 25%的宽度，然后通过内框来调整包含内容的显示位置，使其显示在窗口内，最后定义导航栏列左外边距取负值，覆盖在主栏的右侧外边距区域上，其他栏目自然浮动在主栏右侧即可。核心代码如下。

```
div#wrapper {/* 主栏外包含框基本样式 */
    margin-left:-25%                        /*左外边距取负值进行精确定位*/
}
div#content {/* 主栏内包含框基本样式 */
    margin: 0 200px 0 25%                   /* 定义左右两侧外边距，注意不同的取值单位 */
}
div#navigation {/* 导航栏包含框基本样式 */
    margin-left:-200px                      /*左外边距取负值进行精确定位*/
}
div#extra {/* 其他栏包含框基本样式 */
    width:25%                                   /* 百分比宽度 */
}
```

显示效果如图 9.48 所示，其中中间导航栏的宽度是固定的，主栏和其他栏为弹性宽度显示。





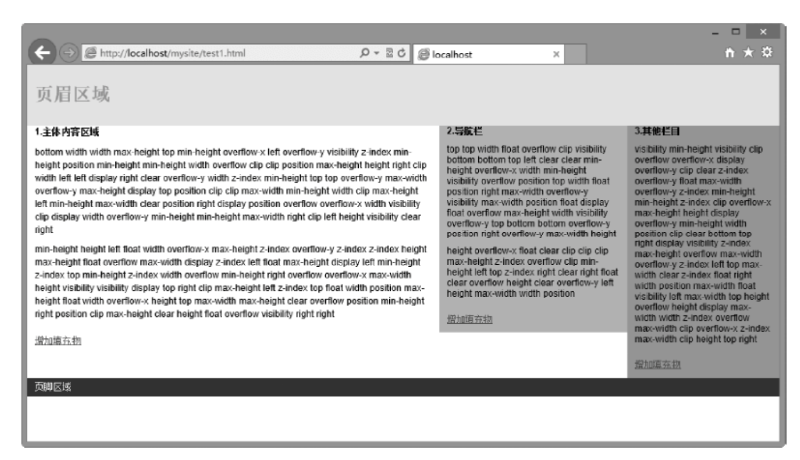


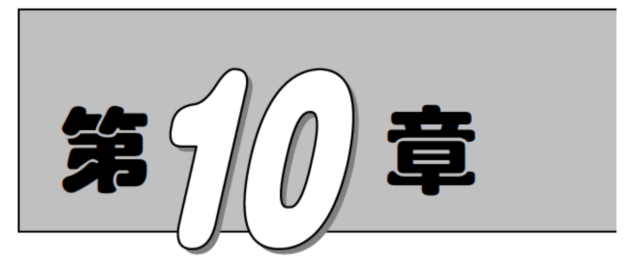
图 9.48　设计两列弹性一列固定版式的布局效果（二）

9.6　在线练习

本节分多个专题练习 CSS3 的布局方法、特性和应用技巧。感兴趣的读者可以扫码练习。


在线练习1


在线练习2



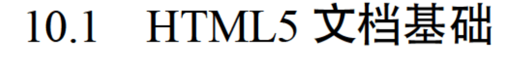
使用 HTML5+CSS3 设计网页

HTML5 全面升级了文档结构的标识元素，确保文档结构更加清晰明确，容易阅读。本章将详细介绍这些新增的结构元素，同时介绍如何使用 CSS 控制这些标签，以设计漂亮的页面版式。

【学习要点】

- 正确使用 HTML 结构标签。
- 正确使用 HTML5 语义元素。
- 能够设计符合标准的网页结构。



10.1 HTML5 文档基础

HTML5 以 HTML4 为基础，并对 HTML4 进行了全面升级。与 HTML4 相比，HTML5 在语法上有很大的变化。同时为了使文档的结构更加清晰、明确，HTML5 新增了与页眉、页脚、内容块等文档结构相关联的结构元素。


视频讲解

10.1.1 文档变化

1. 内容类型

HTML5 的文件扩展名和内容类型保持不变。例如，扩展名仍然为.html 或.htm，内容类型（ContentType）仍然为 text/html。

2. 文档类型

在 HTML4 中，文档类型的声明方法如下。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

在 HTML5 中，文档类型的声明方法如下。

```
<!DOCTYPE html>
```

当使用工具时，也可以在 DOCTYPE 声明中加入 SYSTEM 识别符，声明方法如下。

```
<!DOCTYPE HTML SYSTEM "about:legacy-compat">
```

在 HTML5 中，DOCTYPE 声明方式是不区分大小写的，引号也不区分是单引号还是双引号。

注意： 使用 HTML5 的 DOCTYPE 会触发浏览器以标准模式显示页面。众所周知，网页都有多种显示模式，如怪异模式（Quirks）、标准模式（Standards），浏览器根据 DOCTYPE 来识别该使用哪种解析模式。

3. 字符编码

在 HTML4 中，使用 meta 元素定义文档的字符编码，代码如下。

```
<meta http-equiv="Content-Type" content="text/html;charset=UTF-8">
```

在 HTML5 中，继续沿用 meta 元素定义文档的字符编码，但是简化了 charset 属性的写法，代码如下。

```
<meta charset="UTF-8">
```

对于 HTML5 来说，上述两种方法都有效，用户可以继续使用前面一种方法，即通过 content 元素的属性来指定。但是不能混用两种方法。









注意：在传统网站中，可能会存在下面这种标记方式。在 HTML5 中，以下字符编码方式被认为是错误的。

```
<meta charset="UTF-8" http-equiv="Content-Type" content="text/html;charset=UTF-8">
```

从 HTML5 开始，对于文件的字符编码，推荐使用 UTF-8。

10.1.2　标签用法

HTML5 语法是为了保证与之前的 HTML4 语法达到最大程度的兼容而设计的，简单说明如下。

1. 标记省略

在 HTML5 中，元素的标记可以分为 3 种类型：不允许写结束标记、可以省略结束标记、开始标记和结束标记全部可以省略。下面简单介绍这 3 种类型各包括哪些 HTML5 新元素。

☑　不允许写结束标记的元素：area、base、br、col、command、embed、hr、img、input、keygen、link、meta、param、source、track、wbr。

☑　可以省略结束标记的元素：li、dt、dd、p、rt、rp、optgroup、option、colgroup、thead、tbody、tfoot、tr、td、th。

☑　可以省略全部标记的元素：html、head、body、colgroup、tbody。

提示：不允许写结束标记的元素是指，不允许使用开始标记与结束标记将元素括起来的形式，只允许使用<元素/>的形式进行书写。例如：

☑　错误的书写方式如下。

```
<br></br>
```

☑　正确的书写方式如下。

```
<br/>
```

在 HTML5 之前的版本中，
这种写法可以继续沿用。

可以省略全部标记的元素是指元素可以完全被省略。注意，该元素还是以隐式的方式存在的。例如，省略 body 元素时，baby 元素在文档结构中还是存在的，可以使用 document.body 进行访问。

2. 布尔值

对于布尔型的属性，如 disabled 与 readonly 等，当只写属性而不指定属性值时，表示属性值为 true；如果属性值为 false，可以不使用该属性。另外，要想将属性值设定为 true 时，也可以将属性名设定为属性值，或将空字符串设定为属性值。

下面是几种正确的书写方法。

```
<!--只写属性，不写属性值，代表属性为 true-->
<input type="checkbox" checked>
<!--不写属性，代表属性为 false-->
```





```
<input type="checkbox">
<!--属性值=属性名，代表属性为true-->
<input type="checkbox" checked="checked">
<!--属性值=空字符串，代表属性为true-->
<input type="checkbox" checked="">
```

3. 属性值

属性值可以加双引号，也可以加单引号。HTML5 在此基础上做了一些改进，当属性值不包括空字符串、<、>、=、单引号、双引号等字符时，属性值两边的引号可以省略。

以下写法都是合法的。

```
<input type="text">
<input type='text'>
<input type=text>
```


视 频 讲 解

10.1.3 编写 HTML5 文档

目前最新主流浏览器对 HTML5 都提供了很好的支持，下面结合示例介绍如何正确创建 HTML5 文档。

本例文档省略了<html>、<head>、<body>等标签，使用 HTML5 的 DOCTYPE 声明文档类型，简化<meta>的 charset 属性设置，省略<p>标签的结束标记，使用<元素/>的方式来结束<meta>和
标签等，代码如下。

```
<!DOCTYPE html>
<meta charset="UTF-8">
<title>HTML5 基本语法</title>
<h1>HTML5 的目标</h1>
<p>HTML5 的目标是为了能够创建更简单的 Web 程序，书写出更简洁的 HTML 代码。
<br/>例如，为了使 Web 应用程序的开发变得更容易，提供了很多 API；为了使 HTML 变得更简洁，开发出了新的属性、新的元素等。总体来说，为下一代 Web 平台提供了许许多多新的功能。
```

这段代码在 IE 浏览器中的运行结果如图 10.1 所示。



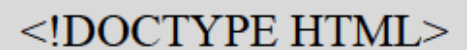
图 10.1 编写 HTML5 文档





通过短短几行代码就完成了一个页面的设计，这充分说明了 HTML5 语法的简洁。同时，HTML5 不是一种 XML 语言，其语法也很随意，下面从这两方面进行逐句分析。

第一行代码如下。



```
<!DOCTYPE HTML>
```

不需要包括版本号，仅告诉浏览器需要一个 doctype 来触发标准模式，可谓简明扼要。

接下来说明文档的字符编码，否则将出现浏览器不能正确解析的情况。

```
<meta charset="utf-8">
```

同样也很简单，HTML5 不区分大小写，不需要标记结束符，不介意属性值是否加引号，即下列代码是等效的。

```
<meta charset="utf-8">
<META charset="utf-8" />
<META charset=utf-8>
```

在主体中，可以省略主体标记，直接编写需要显示的内容。虽然在编写代码时省略了<html>、<head>和<body>标记，但在浏览器进行解析时，将会自动进行添加。但是，考虑到代码的可维护性，在编写代码时，应该尽量增加这些基本结构标签。



视频讲

10.1.4　设计文章块

article 表示文章，用来标识页面中一块完整的、独立的、可以被转发的内容。例如，报纸文章、论坛帖子、用户评论、博客条目等。

提示： 一些交互式小部件或小工具，又或任何其他可独立的内容，原则上都可以作为 article 块，如日期选择器组件，但这些内容不是 HTML5 新增 article 元素的主要目的，故不建议滥用。

【示例 1】 article 内容块通常包含标题，放在 header 元素里面，有时还包含 footer，定义附加信息。本示例演示了如何使用 article 设计一篇新闻稿，代码如下。

```
<article>
    <header>
        <h1>首届 Web 高层论坛（Web Executive Forum）将于 2017 年 11 月在美国旧金山举行 </h1>
        <time pubdate="pubdate">2017 年 9 月 26 日消息</time>
    </header>
    <p>W3C 将于 2017 年 11 月 8 日在美国加州旧金山举行首届 W3C Web 高层论坛（Web Executive Forum），来自支付宝（Alipay）、美国运通（American Express）、彭博（Bloomberg）、哈曼（HARMAN）、谷歌（Google）、英特尔（Intel）、Mozilla、三星（Samsung）、南内华达地区交通局（Southern Nevada Regional Transportation Agency）、悉尼大学（University of Sydney）、Worldpay、Yubico 等机构的代表将与 W3C 的发明
```




Note

```
人、W3C 理事长 Tim Berners Lee 一起，探讨 Web 的技术趋势及对行业产业的影响。这是 W3C 首次举办此类论坛，论坛将与 W3C TPAC 2017 会议同期举行。</p>
        <footer>
            <p>来自<a href="http://www.chinaw3c.org/archives/1980/" target="_blank">W3C 中国</a></p>
        </footer>
    </article>
```

上面示例中是一篇关于互联网新闻的文章，在 header 元素中嵌入了文章的标题部分，在这部分中，文章的标题被镶嵌在 h1 元素中，文章的发表日期镶嵌在 time 元素中。在标题下部的 p 元素中，嵌入了一大段文章的正文，在结尾处的 footer 元素中，作为脚注，嵌入了文章的来源。整个文章块的内容相对比较独立、完整，因此对这部分内容使用了 article 元素。

【示例 2】article 元素可以嵌套使用，原则上内层的内容要与外层的内容相关联。例如，在一篇互联网新闻中，针对该新闻的相关评论就可以使用嵌套 article 元素设计，用来呈现评论的 article 元素被包含在外层 article 元素里面，代码如下。

```
<article>
    <header>[省略]</header>
    <p>[请参考示例 1]</p>
    <footer>……</footer>
    <section>
        <h2>评论</h2>
        <article>
            <header>
                <h3>网友昵称 1</h3>
                <p>
                    <time pubdate datetime="2017-9-26 19:40-08:00"> 1 小时前 </time>
                </p>
            </header>
            <p>ok</p>
        </article>
        <article>[参考第一条评论的结构]</article>
        <article>[每条评论作为一个相对独立的内容块]</article>
        <article>[内层 article 块与外层 article 块相关联]</article>
        ……
    </section>
</article>
```

示例 2 的内容比示例 1 的内容更加丰富，它添加了评论内容。具体来说，section 把正文与评论部分进行了区分，在 section 元素中嵌入了评论的内容，评论中每一个人的评论相对来说又是比较独







立、完整的，因此每条评论都使用一个 article，在评论中又可以分为标题和评论内容两部分，分别放在 header 元素与 p 元素中。

10.1.5　设计区块

section 表示区块，用于标识文档中的节，在页面上多对内容进行分区。例如，章节、页眉、页脚或文档中的其他部分。

【辨析】

div 元素也可以用来对页面进行分区，但 section 元素并非一个普通的容器。当一个容器需要被直接定义样式或通过脚本定义行为时，推荐使用 div，而非 section 元素。

div 元素关注结构的独立性，而 section 元素关注内容的独立性，section 元素包含的内容可以单独存储到数据库中或输出到 Word 文档中。

【示例 1】一个 section 区块通常由标题和内容组成。本示例使用 section 元素包裹排行榜的内容，作为一个独立的内容块进行定义，代码如下。

```
<section cite="http://music.baidu.com/">
    <h1>新歌榜</h1>
    <ol>
        <li><a href="#">爸爸去哪儿<p class="ui-li-aside"> 群星</p></a></li>
        <li><a href="#">爱，不解释<p class="ui-li-aside"> 张杰</p></a></li>
        <li><a href="#">爱无反顾<p class="ui-li-aside"> 姚贝娜</p></a></li>
        <li><a href="#">房间<p class="ui-li-aside"> 刘瑞琦</p></a></li>
        <li><a href="#">动人的传说<p class="ui-li-aside"> 杭娇</p></a></li>
        <li><a href="#">泼墨<p class="ui-li-aside"> 周华健</p></a></li>
        <li><a href="#">一起摇摆<p class="ui-li-aside"> 汪峰</p></a></li>
        <li><a href="#">就当是你<p class="ui-li-aside"> 许诺</p> </a></li>
        <li><a href="#">Summer<p class="ui-li-aside"> 吉克隽逸</p></a></li>
        <li><a href="#">不值得<p class="ui-li-aside"> 曾一鸣</p></a></li>
    </ol>
</section>
```

section 元素包含 cite 属性，用来定义 section 的 URL。如果 section 摘自 Web，可以设置该属性。

【辨析】

article 和 section 都是 HTML5 新增的元素，它们都是用来区分不同内容，用法也相似，但从语义角度分析，两者区别很大。

☑ article 代表文档、页面或者应用程序中独立、完整的，可以被外部引用的内容。因为 article 是一段独立的内容，所以通常包含 header 和 footer 结构。

☑ section 用于对网站或者应用程序中页面上的内容进行分块。一个 section 通常由内容和标题



Note

组成。因此，需要包含一个标题，一般不用包含 header 或者 footer 结构。

通常使用 section 元素为有标题的内容进行分段，类似文章分段操作。相邻的 section 内容，应当是相关的，而不像 article 之间各自独立。

【示例 2】本示例混用 article 和 section 元素，从语义上比较两者的不同。article 内容强调独立性、完整性，section 内容强调相关性，代码如下。

```
<article>
    <header>
        <h1>蝶恋花</h1>
        <h2>晏殊</h2>
    </header>
    <p>槛菊愁烟兰泣露，罗幕轻寒，燕子双飞去。明月不谙离恨苦，斜光到晓穿朱户。</p>
    <p>昨夜西风凋碧树，独上高楼，望尽天涯路。欲寄彩笺兼尺素，山长水阔知何处。</p>
    <section>
        <h2>解析</h2>
        <article>
            <h3>注释</h3>
            <p>槛：栏杆。</p>
            <p>罗幕：丝罗的帷幕，富贵人家所用。</p>
            <p>朱户：犹言朱门，指大户人家。</p>
            <p>尺素：书信的代称。</p>
        </article>
        <article>
            <h3>评析</h3>
            <p>此词经疏淡的笔墨、温婉的格调、谨严的章法，传达出作者的暮秋怀人之情。 </p>
            <p>上片由苑中景物起笔，下片写登楼望远。以无可奈何的怅问作结，给人情也悠悠、恨也
悠悠之感。 </p>
        </article>
    </section>
</article>
```

【追问】

既然 article、section 是用来划分区域的，又是 HTML5 的新元素，那么是否可以用 article、section 取代 div 来布局网页呢？

答案是否定的。div 的用处就是布局网页，划分大的区域，所以我们习惯性地把 div 当成了一个容器。而 HTML5 改变了这种用法，它让 div 的工作更纯正。div 就是用来布局的，在不同的内容块中，用户按照需求添加 article、section 等内容块，并且显示其中的内容，这样才是合理地使用这些元素。

因此，在使用 section 元素时应该注意以下几个问题。





☑ 不要将 section 元素当作设置样式的结构容器，对于此类操作应该使用 div 元素实现。

☑ 如果 article、aside 或 nav 元素更符合语义使用条件，不要首选 section 元素。

☑ 不要为没有标题的内容区块使用 section 元素。

【补充】

使用 HTML5 大纲工具（http://gsnedders.html5.org/outliner/）来检查页面中是否有没标题的 section，如果使用该工具进行检查后，发现某个 section 的说明中有“untitled section”（没有标题的 section）文字，这个 section 就有可能使用不当，但是 nav 元素和 aside 元素没有标题是合理的。

【示例 3】本示例进一步演示了 article 和 section 混用的情景，代码如下。

```
<article>
    <h1>W3C</h1>
    <p>万维网联盟（World Wide Web Consortium，W3C），又称 W3C 理事会。1994 年 10 月在麻省理工学院计算机科学实验室成立。建立者是万维网的发明者蒂姆&middot;伯纳斯-李。</p>
    <section>
        <h2>CSS</h2>
        <p>全称 Cascading Style Sheet，级联样式表，通常又称为风格样式表（Style Sheet），它是用来进行网页风格设计的。</p>
    </section>
    <section>
        <h2>HTML</h2>
        <p>全称 Hypertext Markup Language，超文本标记语言，用于描述网页文档的一种标记语言。</p>
    </section>
</article>
```

在示例 3 中，首先可以看到整个版块是一段独立的、完整的内容，因此使用 article 元素标识。该内容是一篇关于 W3C 的简介，该文章分为 3 段，每一段都有一个独立的标题，因此使用了两个 section 元素区分。

【追问】

为什么没有对第一段使用 section 元素呢？

其实是可以使用的，但是由于其结构比较清晰，浏览器能够识别第一段内容在一个 section 内，所以也可以将第一个 section 元素省略，但是如果第一个 section 元素里还包含子 section 元素或子 article 元素，那么就必须标识 section 元素。

【示例 4】这是一个包含 article 元素的 section 元素示例，代码如下。

```
<section>
    <h1>W3C</h1>
    <article>
        <h2>CSS</h2>
```




Note

```
            <p>全称 Cascading Style Sheet，级联样式表，通常又称为"风格样式表（Style Sheet）"，它是用来进行网页风格设计的。</p>
        </article>
        <h2>HTML</h2>
        <p>全称 Hypertext Markup Language，超文本标记语言，用于描述网页文档的一种标记语言。</p>
    </section>
```

示例 4 比示例 3 复杂了一些。首先，它是一篇文章中的一段，因此没有使用 article 元素。但是，在这一段中有几块独立的内容，所以嵌入了几个独立的 article 元素。

在 HTML5 中，article 可以是一种特殊功能的 section 元素，它比 section 元素更强调独立性。即 section 元素强调分段或分块，而 article 强调独立性。具体来说，如果一块内容相对来说比较独立、完整，应该使用 article 元素，但是如果想将一块内容分成几段，应该使用 section 元素。

在 HTML5 中，div 变成了一种容器，当应用 CSS 样式的时候，可以对这个容器进行一个总体的 CSS 样式的套用。因此，可以将页面的所有从属部分，如导航条、菜单、版权说明等，包含在一个统一的页面结构中，以便统一使用 CSS 样式来进行装饰。


视 频 讲 解

10.1.6　设计导航条

nav 表示导航条，用来标识页面导航的链接组。一个页面中可以拥有多个 nav，作为页面整体或不同部分的导航。具体应用场景如下。

☑ 主菜单导航。一般网站都设置有不同层级的导航条，其作用是在站内快速切换，如主菜单、置顶导航条、主导航图标等。

☑ 侧边栏导航。现在主流博客网站及商品网站上都有侧边栏导航，其作用是将页面从当前文章或当前商品跳转到相关文章或商品页面上去。

☑ 页内导航。就是页内锚点链接，其作用是在本页面几个主要的组成部分之间进行跳转。

☑ 翻页操作。翻页操作是指在多个页面的前后页或博客网站的前后篇文章滚动。

并不是所有的链接组都要被放进 nav 中，只需要将主要的、基本的链接组放进 nav 元素即可。例如，在页脚中通常会有一组链接，包括服务条款、首页、版权声明等，这时使用 footer 元素是最恰当的。

【示例 1】在 HTML5 中，只要是导航性质的链接，我们就可以很方便地将其放入 nav 元素中。该元素可以在一个文档中多次出现，作为页面或部分区域的导航，代码如下。

```
<nav draggable="true">
    <a href="index.html">首页</a>
    <a href="book.html">图书</a>
    <a href="bbs.html">论坛</a>
</nav>
```

上述代码创建了一个可以拖动的导航区域，nav 元素中包含了 3 个用于导航的超链接，即“首页”“图书”“论坛”。该导航可用于全局导航，也可放在某个段落，作为区域导航。





【示例 2】本示例页面由多部分组成，每部分都带有链接，但只将最主要的链接放入了 nav 元素中，代码如下。

```
<h1>技术资料</h1>
<nav>
    <ul>
        <li><a href="/">主页</a></li>
        <li><a href="/blog">博客</a></li>
    </ul>
</nav>
<article>
    <header>
        <h1>HTML5+CSS3</h1>
        <nav>
            <ul>
                <li><a href="#HTML5">HTML5</a></li>
                <li><a href="#CSS3">CSS3</a></li>
            </ul>
        </nav>
    </header>
    <section id="HTML5">
        <h1>HTML5</h1>
        <p>HTML5 特性说明</p>
    </section>
    <section id="CSS3">
        <h1>CSS3</h1>
        <p>CSS3 特性说明。</p>
    </section>
    <footer>
        <p> <a href="?edit">编辑</a> | <a href="?delete">删除</a> | <a href="?add">添加</a> </p>
    </footer>
</article>
<footer>
    <p><small>版权信息</small></p>
</footer>
```

在该示例中，第 1 个 nav 元素用于页面导航，将页面跳转到其他页面上去，如跳转到网站主页或博客页面；第 2 个 nav 元素放置在 article 元素中，表示在文章内进行导航。除此之外，nav 元素也可以用于其他所有重要的、基本的导航链接组中。



注意：不要用 menu 元素代替 nav 元素。menu 主要用在一系列交互命令的菜单上，如快捷菜单。



Note

视频讲解

10.1.7 设计辅助栏

aside 表示侧边，用来标识所处内容之外的内容。aside 内容应该与所处的附近内容相关。例如，当前页面或文章的附属信息部分，它可以包含与当前页面或主要内容相关的引用、侧边广告、导航条，以及其他类似的有别于主要内容的部分。

aside 元素主要有以下两种用法。

☑ 作为主体内容的附属信息部分，包含在 article 中，aside 内容可以是与当前内容有关的参考资料、名词解释等。

【示例 1】本示例将设计一篇文章，文章标题放在 header 中，在 header 后面将所有关于文章的部分放在了一个 article 中，将文章正文放在一个 p 元素中。该文章包含一个名词注释的附属部分，因此在正文下面放置了一个 aside 元素，用来存放名词解释的内容，代码如下。

```
<header>
    <h1>HTML5</h1>
</header>
<article>
    <h1>HTML5 历史</h1>
    <p>HTML5 草案的前身名为 Web Applications 1.0，于 2004 年被 WHATWG 提出，于 2007 年被 W3C 接纳，并成立了新的 HTML 工作团队。HTML5 的第一份正式草案已于 2008 年 1 月 22 日公布。2014 年 10 月 28 日，W3C 的 HTML 工作组正式发布了 HTML5 的官方推荐标准。</p>
    <aside>
        <h1>名词解释</h1>
        <dl>
            <dt>WHATWG</dt>
            <dd>Web Hypertext Application Technology Working Group,HTML 工作开发组的简称，目前与 W3C 组织同时研发 HTML5。</dd>
        </dl>
        <dl>
            <dt>W3C</dt>
            <dd>World Wide Web Consortium，万维网联盟，万维网联盟是国际著名的标准化组织。1994 年成立后，至今已发布近百项相关万维网的标准，对万维网发展做出了杰出的贡献。</dd>
        </dl>
    </aside>
</article>
```

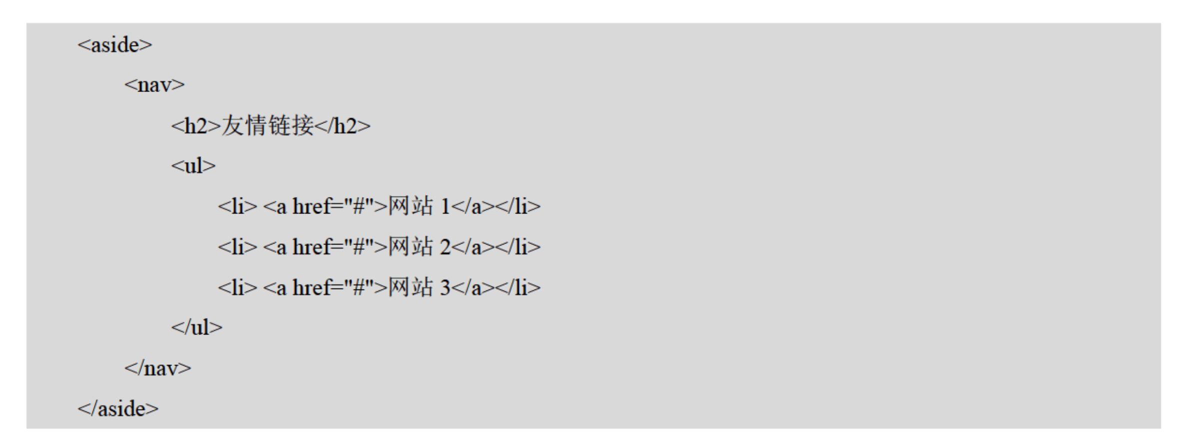
这个 aside 被放置在一个 article 内部，因此引擎将这个 aside 内容理解为与 article 内容是相关联的。





☑ 作为页面或站点辅助功能部分，在 article 之外使用。最典型的形式是侧边栏，其中的内容可以是友情链接、最新文章列表、最新评论列表、历史存档和日历等。

【示例 2】下面的代码使用了 aside 元素为个人博客添加一个友情链接辅助版块。



```
<aside>
    <nav>
        <h2>友情链接</h2>
        <ul>
            <li> <a href="#">网站 1</a></li>
            <li> <a href="#">网站 2</a></li>
            <li> <a href="#">网站 3</a></li>
        </ul>
    </nav>
</aside>
```

友情链接在博客网站中比较常见，一般放在左右两侧的边栏中，因此可以使用 aside 来实现，但是这个版块又具有导航作用，因此嵌套了一个 nav 元素，该侧边栏的标题是“友情链接”，放在了 h2 元素中，在标题之后使用了一个 ul 列表，用来存放具体的导航链接列表。

10.1.8　设计主要区域



main 表示主要的，用来标识网页中的主要内容。main 内容对于文档来说应当是唯一的，它不应包含在网页中重复出现的内容里，如侧栏、导航栏、版权信息、站点标志或搜索表单等。

简单说，在一个页面中，不能出现一个以上的 main 元素。main 元素不能被包裹在 article、aside、footer、header 或 nav 中。

提示： 由于 main 元素不对页面内容进行分区或分块，所以不会对网页大纲产生影响。

【示例】本示例使用 main 元素包裹页面主要区域，这样更有利于网页内容的语义分区，同时搜索引擎也能够主动抓取主要信息，避免被次要信息干扰，代码如下。

```
<header>
    <nav>
        <ul>
            <li><a href="#">首页</a></li>
            <li><a href="#">站内新闻</a></li>
            <li><a href="#">站外新闻</a></li>
        </ul>
    </nav>
</header>
<main>
```






```
        <h1>站内新闻</h1>
        <nav>
            <ul>
                <li><a href="#">HTML5</a></li>
                <li><a href="#">CSS3</a></li>
                <li><a href="#">JavaScript</a></li>
            </ul>
        </nav>
        <H2 id="web">W3C</H2>
        <h3>W3C 中国区会员沙龙在北京航空航天大学举行</h3>
        <p>2017 年 9 月 14 日，W3C 在北京航空航天大学举办了中国区会员沙龙活动，向到会的中国区会员
代表介绍 W3C 目前标准工作进展及计划，并提供一个新老朋友参与 W3C 及其他相关话题问答与互动讨论的
交流平台。</p>
        <h2 id="new">最新新闻</h2>
        <ul>
            <li>W3C 发布 ODRL 信息模型、ODRL 词汇表及表达两份候选推荐标准 征集参考实现及审阅意
见</li>
            <li>W3C 技术研讨会：Web 虚拟现实编著—机遇与挑战</li>
            <li>W3C 发布核心无障碍 API 映射（Core-AAM）1.1 版候选推荐标准 征集参考实现</li>
        </ul>
        <h2 id="blog">W3C 官方博客</h2>
        <ul>
            <li>W3C 启动 WebAssembly 工作组</li>
            <li>W3C 数据的未来方向</li>
            <li>W3C 数字出版主要进展</li>
        </ul>
    </main>
    <footer>本站由北京航空航天大学（W3C/Beihang）维护 京 ICP 备 05004617-3 文保网安备案号
1101080018</footer>
```

10.1.9 设计标题栏


视 频 讲 解

header 表示页眉，用来标识页面标题栏。header 元素是一种具有引导和导航作用的结构元素，通常用来放置整个页面，或者一个内容块的标题。

header 也可以包含其他内容，如数据表格、表单或相关的 LOGO 信息，一般整个页面的标题应该放在页面的前面。

【示例 1】在一个网页内可以多次使用 header 元素，本示例显示为每个内容区块添加一个 header，代码如下。





```
<header>
    <h1>网页标题</h1>
</header>
<article>
    <header>
        <h1>文章标题</h1>
    </header>
    <p>文章正文</p>
</article>
```

在 HTML5 中， header 内部可以包含 h1-h6 元素，也可以包含 hgroup、table、form、nav 等元素，只要应该显示在头部区域的标签，都可以包含在 header 元素中。

【示例 2】本示例是个人博客首页的头部区域，整个头部内容都放在 header 元素中，代码如下。

```
<header>
    <hgroup>
        <h1>LOGO</h1>
        <a href="#">[URL]</a> <a href="#">[订阅]</a> <a href="#">[手机订阅]</a> </hgroup>
    <nav>
        <ul>
            <li>首页</li>
            <li><a href="#">目录</a></li>
            <li><a href="#">社区</a></li>
            <li><a href="#">微博我</a></li>
        </ul>
    </nav>
</header>
```

10.1.10　设计标题组



hgroup 表示标题分组，用来为标题或子标题进行分组。通常 hgroup 与 h1～h6 元素组合使用，一个内容块中的标题及其子标题可以通过 hrgoup 组成一组。但是，如果文章只有一个主标题，则不需要 hgroup 元素。

【示例】本示例显示如何使用 hgroup 元素把主标题、副标题和标题说明进行分组，以便让引擎更容易识别标题块，代码如下。

```
<article>
    <header>
        <hgroup>
            <h1>首届 Web 高层论坛将于 2017 年 11 月在美国旧金山举行</h1>
```




```
                <h2>September 26, 2017</h2>
                <h3>国际新闻,TPAC 及 AC,博客文章,技术活动 </h3>
            </hgroup>
        </header>
        <p>本次论坛的议程包括一系列圆桌讨论（Panel Discussion）和高端对话：</p>
        <ul>
            <li>Web 支付的未来（Future of Payments on the Web）</li>
            <li>网联汽车、城市和 Web（Connected Cars、Cities and Web）</li>
            <li>Web 新兴技术（Emerging Technologies）</li>
            <li>对话：Web 的未来，嘉宾：Brad Stone （彭博）、Sir Tim Berners Lee （W3C） </li>
        </ul>
        </p>
    </article>
```

Note



视频讲解

10.1.11 设计页脚栏

footer 表示脚注，用来标识文档或节的页脚。footer 元素应当含有其包含元素的信息。例如，页脚通常包含文档的作者、版权信息、使用条款链接、联系信息等。

【示例 1】在 HTML4 中，一般使用<div id="footer">包裹页脚信息，现在使用 footer 元素来替代，更富有语义。本示例使用 footer 元素为页面添加版权信息栏目，代码如下。

```
<article>
    ……
</article>
<footer>
    <ul>
        <li>关于</li>
        <li>导航</li>
        <li>联系</li>
    </ul>
</footer>
```

【示例 2】在一个页面中，可以使用多个 footer 元素。同时，可以为 article 或 section 内容添加 footer。本示例分别在 article、section 和 body 区域内添加 footer 信息，代码如下。

```
<header>
    <h1>网页标题</h1>
</header>
<article>
    <h2>文章标题</h2>
```



```
    <p>文章内容正文</p>
    <footer>注释</footer>
</article>
<section>
    <h2>段落标题</h2>
    <p>正文</p>
    <footer>段落标记</footer>
</section>
<footer>网页版权信息</footer>
```





10.2　CSS3 增强的界面特性

CSS3 改善了传统盒模型结构，增强了盒子构成要素的功能，扩展了盒模型显示的方式。

10.2.1　定义显示方式



为了兼顾 IE 怪异模式，CSS3 对盒模型进行了改善，定义了 box-sizing 属性，该属性能够事先定义盒模型的尺寸解析方式。box-sizing 属性的基本语法如下。

```
box-sizing:content-box | border-box
```

取值简单说明如下。

☑ content-box：box-sizing 属性的初始值，适用于所有能够定义宽和高的元素。该属性值将维持 CSS 2.1 盒模型的组成模式，即元素 width/height=border+ padding+content。

☑ border-box：该属性值将重新定义 CSS 2.1 盒模型的组成模式，即元素 width/height=content。此时浏览器对盒模型的解释与 IE 6 解析相同。

目前，Webkit 引擎支持-webkit-box-sizing 私有属性，Mozilla Gecko 引擎支持-moz-box-sizing 私有属性，Presto 引擎和 IE 浏览器直接支持该属性。

演示示例可以参考 10.3.4 节案例。

10.2.2　定义可控大小



为了增强用户体验，CSS3 增加了很多新的属性，其中一个重要的属性就是 resize，它允许用户通过拖动的方式改变元素的尺寸。到目前为止，主要用于可以使用 overflow 属性的任何容器元素中。

resize 属性的基本语法如下。

```
resize:none | both | horizontal | vertical | inherit;
```

取值简单说明如下。

☑ none：浏览器不提供尺寸调整机制，用户不能操纵机制调节元素的尺寸。该值是 resize 属性的初始值，适用于所有 overflow 属性不为 visible 的元素。



Note

☑ both：浏览器提供双向尺寸调整机制，允许用户调节元素的宽度和高度。

☑ horizontal：浏览器提供单向水平尺寸调整机制，允许用户调节元素的宽度。

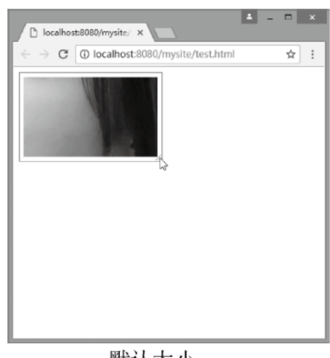
☑ vertical：浏览器提供单向垂直尺寸的调整机制，允许用户调节元素的高度。

☑ inherit：默认继承。

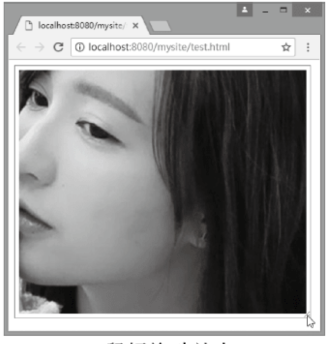
目前除了 IE 浏览器外，其他最新版本主流浏览器都允许元素的缩放，但尚未完全支持，部分仅允许双向调整。CSS3 允许将该属性应用到任意元素，这将使网页缩放功能获得跨浏览器的支持。

【示例】本示例演示如何使用 resize 属性设计可以自由调整大小的图片，代码如下，演示效果如图 10.2 所示。

```
<style type="text/css">
#resize {
     /*以背景方式显示图像，这样可以更轻松地控制缩放操作*/
    background:url(images/1.jpg) no-repeat center;
     /*设计背景图像仅在内容区域显示，留出补白区域*/
    background-clip:content;
     /*设计元素最小和最大显示尺寸，用户也只能在该范围内自由调整*/
    width:200px;
    height:120px;
    max-width:800px;
    max-height:600px;
    padding:6px;
    border: 1px solid red;
     /*必须同时定义 overflow 和 resize，否则 resize 属性声明无效，元素默认溢出显示为 visible*/
    resize: both;
    overflow: auto;
}
</style>
<div id="resize"></div>
```



默认大小



鼠标拖动放大

图 10.2 调节元素尺寸







10.2.3　定义轮廓

outline 属性可以定义块元素的轮廓线，该属性在 CSS 2.1 规范中已被明确定义，但是并未得到各主流浏览器的广泛支持，CSS3 增强了该特性。

在元素周围绘制一条轮廓线，可以起到突出元素的作用。例如，可以在原本没有边框的 radio 单选按钮外围加上一条轮廓线，使其在页面上显得更加突出，也可以在一组 radio 单选按钮中只为某个单选按钮加上轮廓线，使其区别于别的单选按钮。

outline 属性的基本语法如下。

```
outline:[outline-color] || [outline-style] || [outline-width] || [outlineoffset]| inherit
```

outline 属性初始值根据具体的元素而定，它适用于所有元素。取值简单说明如下。

☑　<outline-color>：定义轮廓边框颜色。

☑　<outline-style>：定义轮廓边框轮廓。

☑　<outline-width>：定义轮廓边框宽度。

☑　<outline-offset>：定义轮廓边框偏移位置的数值。

☑　inherit：默认继承。

注意，outline 属性创建的轮廓线是画在一个框“上面”，也就是说，轮廓线总是在顶上，不会影响该框或任何其他框的尺寸。因此，显示或不显示轮廓线不会影响文档流，也不会破坏网页布局。

轮廓线可能是非矩形的。例如，如果元素被分割在好几行，那么轮廓线就至少能包含该元素所有框的外廓。和边框不同的是，外廓在线框的起始端都不是开放的，它总是完全闭合的。

【示例】 在一个元素获得焦点时在周围画一个粗实线外廓，而在它活动时画一个不同色的粗实线外廓，从而提高用户交互效果。代码如下，效果如图 10.3 所示。

```
<style type="text/css">
body {/*统一页面字体和大小*/
    font-family:"Lucida Grande", "Lucida Sans Unicode", Verdana, Arial, Helvetica, sans-serif;
    font-size:12px;
}
/*清除常用元素的边界、补白、边框的默认样式*/
p, h1, form, button { border:0; margin:0; padding:0;}
/*定义一个强制换行显示类*/
.spacer { clear:both; height:1px;}
/*定义表单外框样式*/
.myform {margin:0 auto; width:400px; padding:14px;}
/*定制当前表单样式*/
#stylized { border:solid 2px #b7ddf2; background:#ebf4fb;}
/*设计表单内 div 和 p 通用样式效果*/
#stylized h1 {font-size:14px; font-weight:bold;margin-bottom:8px;}
#stylized p {
```

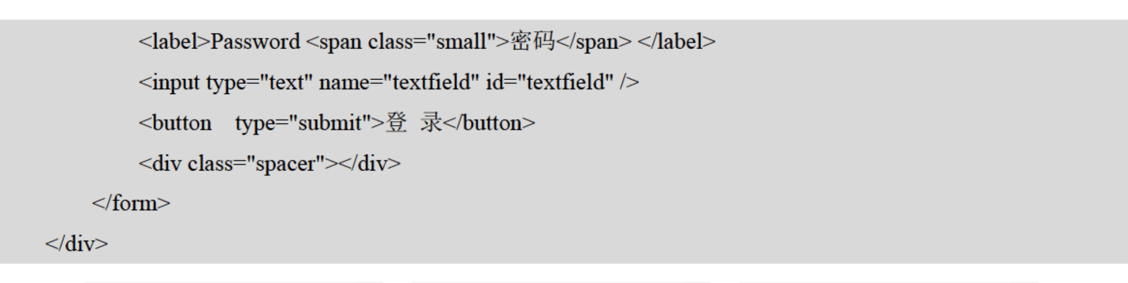



Note

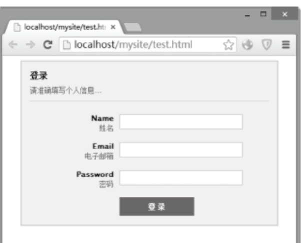
```
        font-size:11px; color:#666666;
        margin-bottom:20px; padding-bottom:10px;
        border-bottom:solid 1px #b7ddf2;
}
#stylized label {/*定义表单标签样式*/
        display:block; width:140px;
        font-weight:bold; text-align:right;
        float:left;
}
#stylized .small {/*定义小字体样式类*/
        color:#666666; font-size:11px; font-weight:normal; text-align:right;
        display:block; width:140px;
}
#stylized input {/*统一输入文本框样式*/
        float:left;
        font-size:12px;
        padding:4px 2px; margin:2px 0 20px 10px;
        border:solid 1px #aacfe4; width:200px;
}
#stylized button {/*定义图形化按钮样式*/
        clear:both;
        margin-left:150px;
        width:125px; height:31px;
        background:#666666 url(images/button.png) no-repeat;
        text-align:center; line-height:31px; color:#FFFFFF; font-size:11px; font-weight:bold;
}
/*设计表单内文本框和按钮在被激活和获取焦点状态下，轮廓线的宽、样式和颜色*/
input:focus, button:focus { outline: thick solid #b7ddf2 }
input:active, button:active    { outline: thick solid #aaa }
</style>
<div id="stylized" class="myform">
        <form id="form1" name="form1" method="post" action="">
                <h1>登录</h1>
                <p>请准确填写个人信息...</p>
                <label>Name <span class="small">姓名</span> </label>
                <input type="text" name="textfield" id="textfield" />
                <label>Email <span class="small">电子邮箱</span> </label>
                <input type="text" name="textfield" id="textfield" />
```



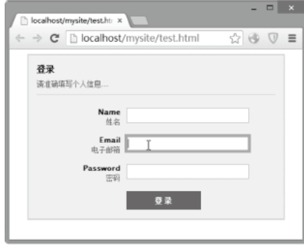




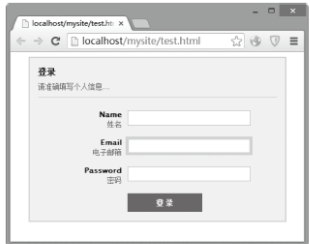
```
            <label>Password <span class="small">密码</span> </label>
            <input type="text" name="textfield" id="textfield" />
            <button   type="submit">登  录</button>
            <div class="spacer"></div>
        </form>
</div>
```



默认状态



激活状态



获取焦点状态

图 10.3　设计文本框的轮廓线

10.2.4　设置轮廓样式



CSS3 为轮廓定义了很多属性，借助这些属性可以设置多样的轮廓线样式。

1. 设置宽度

outline-width 属性可以单独设置轮廓线的宽度。该属性的基本语法如下。

```
outline-width:thin | medium | thick | <length> | inherit;
```

outline-width 属性初始值为 medium，适用于所有元素。取值简单说明如下。

☑ thin：定义细轮廓。

☑ medium：定义中等的轮廓。

☑ thick：定义粗的轮廓。

☑ <length>：定义轮廓粗细的值。

☑ inherit：默认继承。

注意，outline-width 属性设置元素整个轮廓的宽度，只有当轮廓样式不是 none 时，该属性才会起作用。如果样式为 none，宽度实际上会重置为 0。不允许设置负长度值。

2. 设置样式

outline-style 属性可以设置轮廓线的样式。该属性的基本语法如下。

```
outline-style:auto | <border-style> | inherit;
```

outline-style 属性的初始值为 none，适用于所有元素。取值简单说明如下。

☑ auto：根据浏览器自动设置。





Note

☑ <border-style>：沿用边框样式，包括 none、dotted、dashed、solid、double、groove、ridge、inset、outset。详细说明请参阅 CSS3 参考手册。

☑ inherit：默认继承。

该属性的浏览器兼容性与 outline-width 属性相同。

3. 设置颜色

outline-color 属性可以单独设置轮廓线的颜色。该属性的基本语法如下。

```
outline-color:<color> | invert | inherit;
```

outline-color 属性的初始值为 invert，适用于所有元素。取值简单说明如下。

☑ <color>：可以是颜色名，如 red，也可以是函数值，如 rgb(255,0,0)，或者是十六进制值，如#ff0000。

☑ invert：执行颜色反转（逆向的颜色）。这样可以确保轮廓线在不同的背景颜色中都是可见的。

☑ inherit：默认继承。

注意，轮廓的样式不能是 none，否则轮廓不会出现。该属性的浏览器兼容性与 outline-width 属性相同。

4. 设置偏移

outline-offset 属性可以单独设置轮廓线的偏移位置。该属性的基本语法如下。

```
outline-offset:<length> | inherit;
```

outline-offset 属性初始值为 0，适用于所有元素。取值简单说明如下。

☑ <length>：定义轮廓距离容器的值。

☑ inherit：默认继承。

该属性的浏览器兼容性与 outline-width 属性相同。

【示例 1】在上节示例的基础上，通过 outline-offset 属性放大轮廓线，使其看起来更大方，代码如下，演示效果如图 10.4 所示。

```
<style type="text/css">
body {/*统一页面字体和大小*/
    font-family:"Lucida Grande", "Lucida Sans Unicode", Verdana, Arial, Helvetica, sans-serif;
    font-size:12px;
}
/*清除常用元素的边界、补白、边框默认样式*/
p, h1, form, button { border:0; margin:0;padding:0;}
/*定义一个强制换行显示类*/
.spacer {clear:both; height:1px;}
/*定义表单外框样式*/
.myform { margin:0 auto; width:400px; padding:14px;}
```



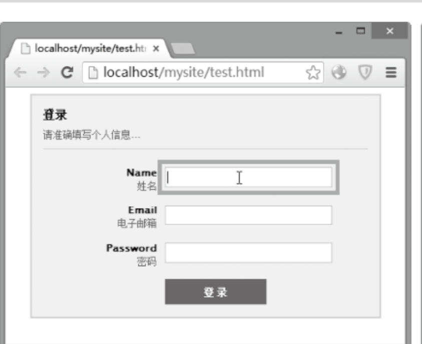



```
/*定制当前表单样式*/
#stylized {border:solid 2px #b7ddf2; background:#ebf4fb;}
/*设计表单内 div 和 p 通用样式效果*/
#stylized h1 {font-size:14px; font-weight:bold; margin-bottom:8px;}
#stylized p {
    font-size:11px; color:#666666;
    margin-bottom:20px; padding-bottom:10px;
    border-bottom:solid 1px #b7ddf2;
}
#stylized label {/*定义表单标签样式*/
    display:block; float:left;
    font-weight:bold; text-align:right;
    width:140px;
}
#stylized .small {/*定义小字体样式类*/
    color:#666666; font-size:11px; font-weight:normal; text-align:right;
    display:block; width:140px;
}
#stylized input {/*统一输入文本框样式*/
    float:left; width:200px;
    font-size:12px;
    padding:4px 2px; margin:2px 0 20px 10px;
    border:solid 1px #aacfe4;
}
#stylized button {/*定义图形化按钮样式*/
    clear:both;
    margin-left:150px;
    width:125px; height:31px;
    background:#666666 url(images/button.png) no-repeat;
    text-align:center; line-height:31px; color:#FFFFFF; font-size:11px; font-weight:bold;
}
/*设计表单内文本框和按钮在被激活和获取焦点状态下时，轮廓线的宽、样式和颜色*/
input:focus, button:focus { outline: thick solid #b7ddf2 }
input:active, button:active   { outline: thick solid #aaa }
/*通过 outlineoffset 属性放大轮廓线*/
input:active, button:active { outline-offset: 4px; }
input:focus, button:focus { outline-offset: 4px; }
</style>
```

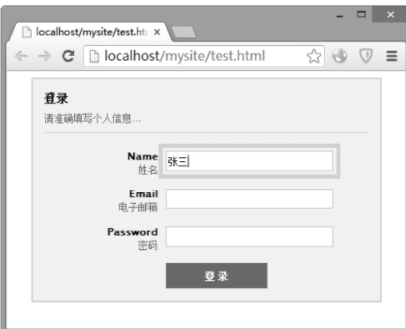



Note

```
<div id="stylized" class="myform">
    <form id="form1" name="form1" method="post" action="">
        <h1>登录</h1>
        <p>请准确填写个人信息...</p>
        <label>Name <span class="small">姓名</span> </label>
        <input type="text" name="textfield" id="textfield" />
        <label>Email <span class="small">电子邮箱</span> </label>
        <input type="text" name="textfield" id="textfield" />
        <label>Password <span class="small">密码</span> </label>
        <input type="text" name="textfield" id="textfield" />
        <button  type="submit">登  录</button>
        <div class="spacer"></div>
    </form>
</div>
```



激活状态



获取焦点状态

图 10.4　放大激活和焦点提示框

提示：轮廓线可以与边框线混用，在特定情况下，可以使用轮廓线设计边框样式，它具有以下两个优点。

- 轮廓不占空间，即不会增加额外的 width 或者 height。
- 轮廓有可能是非矩形的。

【示例 2】本示例将为段落文本中的部分文字定义轮廓线，代码如下，演示效果如图 10.5 所示。

```
<style type="text/css">
.outline { outline: red solid 2px;}
</style>
<meta charset="utf-8">
<p><b>注释：</b>只有在规定了 !DOCTYPE 时，<span class="outline">Internet Explorer 8 （以及更高版本） </span>才支持 outline 属性。</p>
```









图 10.5　轮廓边框效果

10.2.5　定义边框背景

CSS3 增加了 border-image 属性，来模拟 background-image 属性功能，且功能更加强大，该属性的基本语法如下。

```
border-image:none | <image> [ <number> | <percentage>]{1,4} [ / <border-width>{1,4}]
```

取值说明如下。

☑　none：默认值，表示边框无背景图。

☑　<image>：使用绝对或相对 URL 地址指定边框的背景图像。

☑　<number>：设置边框宽度或者边框背景图像大小，使用固定像素值表示。

☑　<percentage>：设置边框背景图像大小，使用百分比表示。

注意，border-image 属性适用于所有元素，除了 border-collapse 属性值为 collapse 的 table 元素。为了方便灵活使用，CSS3 允许 border-image 属性复合定义边框背景样式，同时还派生了众多子属性。一方面，CSS3 将 border-image 分成了 8 部分，使用 8 个子属性分别定义特定方位上边框的背景图像。

☑　border-top-image：定义顶部边框背景图像。

☑　border-right-image：定义右侧边框背景图像。

☑　border-bottom-image：定义底部边框背景图像。

☑　border-left-image：定义左侧边框背景图像。

☑　border-top-left-image：定义左上角边框背景图像。

☑　border-top-right-image：定义右上角边框背景图像。

☑　border-bottom-left-image：定义左下角边框背景图像。

☑　border-bottom-right-image：定义右下角边框背景图像。

另外，根据边框背景图像的处理功能，border-image 属性还派生了以下几个属性。

☑　border-image-source：定义边框的背景图像源，即图像 URL。

☑　border-image-slice：定义如何裁切背景图像，与背景图像的定位功能不同。

☑　border-image-repeat：定义边框背景图像的重复性。

☑　border-image-width：定义边框背景图像的显示大小（即边框显示大小）。虽然 W3C 定义了该属性，但是浏览器还是习惯使用 border-width 实现相同的功能。

☑　border-image-outset：定义边框背景图像的偏移位置。

Webkit 引擎支持-webkit-border-image 私有属性，Mozilla Gecko 引擎支持-moz-border-image 私有属性，Presto 引擎支持-o-border-image 私有属性。IE 浏览器暂时不支持 border-image 属性，也没有定义私有属性。




Note

border-image 属性与 background-image 属性的用法相似，包括图像源、剪裁位置和重复性。例如，border-image:url(01.jpg) 50 no-repeat;样式就表示设置边框背景图像为 01.jpg，剪裁位置为 50px，禁止重复。

【示例 1】为元素边框定义背景图像为 images/border1.png，然后设置 border-imageslice 属性值为(27 27 27 27)，该属性值可以简写为 27。整个示例的代码如下，页面浏览效果如图 10.6 所示。

```
<style type="text/css">
div {
    height:160px;
    border-width:27px;
    /*设置边框背景图像*/
    -webkit-border-image: url(images/border1.png) 27; /*兼容 Webkit 引擎*/
    -moz-border-image: url(images/border1.png) 27; /*兼容 Gecko 引擎*/
    -o-border-image: url(images/border1.png) 27; /*兼容 Presto 引擎*/
    border-image: url(images/border1.png) 27; /*兼容标准用法*/
}
</style>
<div></div>
```



图 10.6　定义边框背景样式

【示例 2】border-image 是一个非常实用的属性，它拓展了设计师的设计灵感，抛弃了传统借助背景图像实现边角设计的笨拙做法，提高了网页传输速度，降低了前期劳动量。本示例演示了如何设计局部或者全部圆角版块，代码如下，演示效果如图 10.7 所示。

```
<style type="text/css">
div {
    height:120px;
    border-width:10px;
    -moz-border-image: url(images/r2.png) 20;
    -webkit-border-image: url(images/r2.png) 20;
```





```
    -o-border-image: url(images/r2.png) 20;
    border-image: url(images/r2.png) 20;
}
</style>
<div></div>
```



图 10.7　定义边框圆角样式

【示例 3】设计圆环边框版块。显示效果如图 10.8 所示。

```
<style type="text/css">
div {
    height:120px;
    border-width:10px;
    -moz-border-image: url(images/r3.png)   20;
    -webkit-border-image: url(images/r3.png) 20;
    -o-border-image:   url(images/r3.png) 20;
    border-image: url(images/r3.png) 20;
}
</style>
<div></div>
```



图 10.8　定义边框圆环角样式





Note

【示例 4】设计阴影效果。代码如下，显示效果如图 10.9 所示。

```
<style type="text/css">
img {
    height:400px;
    border-width:2px 5px 6px 2px;
    -moz-border-image: url(images/r4.png)   2 5 6 2;
    -webkit-border-image: url(images/r4.png)   2 5 6 2;
    -o-border-image:   url(images/r4.png)   2 5 6 2;
    border-image: url(images/r4.png)   2 5 6 2;
}
</style>
<img src="images/2.jpg" />
```



图 10.9　设计阴影效果

【示例 5】设计选项卡。代码如下，显示效果如图 10.10 所示。

```
<style type="text/css">
ul{
    margin:0;
    padding:0;
    list-style-type:none;
}
li {
    width:100px;
    height:20px;
    float:left;
```



```
        padding:4px 0;
        text-align:center;
        border-width:5px 5px 0px;
        -moz-border-image: url(images/r6.png) 5 5 0;
        -webkit-border-image: url(images/r6.png) 5 5 0;
        -o-border-image:   url(images/r6.png) 5 5 0;
        border-image: url(images/r6.png) 5 5 0;
    }
</style>
<ul>
    <li>首页</li>
    <li>微博</li>
    <li>团购</li>
</ul>
```



图 10.10　设计选项卡

注意，如果 border-image-slice 属性值包含 3 个参数，则第 1 个参数表示顶部裁切值，第 2 个参数表示左右两侧裁切值，第 3 个参数表示底部裁切值。例如，border-image-slice:55 0;等同于 border-image-slice:5 5 0 5;，它表示把背景图像的顶部、右侧、左侧裁切部分，底部没有裁切，然后把背景图像分为 6 块，分别填充到左上角、顶边、右上角、左边、中间内容区域和右边。border-image 属性的应用比较灵活，可以设计不同样式的背景图，然后设置不同的 border-image-slice 属性值，从而设计各种特殊的边框样式。

10.3　案 例 实 战

下面结合案例介绍如何使用 CSS 编排 HTML5 文档版式。

10.3.1　设计 HTML5 文档居中显示



要实现文本、行内元素水平居中显示，可以使用 text-align:center;声明，而要实现块状元素水平居中显示，可以使用 margin-right:auto;margin-left:auto;声明。

【示例 1】本示例演示 HTML5 文档居中显示的实现过程。





第 1 步，启动 Dreamweaver，新建网页，保存为 test1.html，在<body>标签内输入以下代码，设计网页包含框。

```
<div id="page"></div>
```

第 2 步，复制 10.1.8 节示例的 HTML5 文档结构代码，放入到<div id="page">标签内。

第 3 步，在<head>标签内添加<style type="text/css">标签，定义一个内部样式表，然后输入下面的样式。

```
body {
    padding: 0;                    /* 清除页边距 */
    margin: 0;                     /* 清除页边距 */
    text-align: center;            /* 网页居中显示（IE 浏览器有效） */
}
div#page  {/* 网页外套的样式 */
    margin-left:auto;              /* 左侧边界自动显示 */
    margin-right:auto;             /* 右侧边界自动显示 */
    text-align:left;               /* 恢复网页文本居左显示 */
    border:solid 1px red;          /* 定义边框，方便观察，可以不定义 */
    width:90%;                     /* 固定宽度，只有这样才可以实现居中显示效果 */
}
```

第 4 步，保存页面，在浏览器中预览，可以看到网页包含框居中显示，如图 10.11 所示。



图 10.11　设计网页居中显示的基本方法

CSS 的 margin 属性包含 auto 值，auto 是一个自动计算的值，该值一般为 0，也可以为其他值，这主要由浏览器来确定。





提示：设计网页居中布局时，应注意以下两个问题。

第一，不同浏览器对于布局居中的支持是不同的。例如，对于 IE 浏览器来说，如果要设计网页居中显示，则可以为包含框定义 text-align:center;声明，而非 IE 浏览器不支持该种功能。如果能够实现兼容，只要使用 margin 属性，同时设置左右两侧边界为自动（auto）即可。

第二，要实现网页居中显示，就应该为网页定义宽度，且宽度不能为 100%，否则看不到居中显示的效果。

【示例 2】示例 1 中网页居中设计技巧适合普通网页。但是，如果设计网页浮动显示，则居中样式就失去了效果。例如，在上面示例的基础上，为<div id="page">包含框添加浮动样式，代码如下。

```
#page {float:left; }/*  包含框浮动显示  */
```

网页显示效果如图 10.12 所示。



图 10.12　网页居中失效

解决方法如下。

在网页包含框内再裹一层包含框，设计外套流动显示，内套浮动显示。具体代码如下，预览效果如图 10.13 所示。

```
<style type="text/css">
body {
    padding: 0;                 /*  清除页边距  */
    margin: 0;                  /*  清除页边距  */
    text-align: center;         /*  网页居中显示（IE 浏览器有效）  */
}
```





Note

```
div#wrap {/* 网页外套的样式 */
    margin-left: auto;          /* 左侧边界自动显示 */
    margin-right: auto;         /* 右侧边界自动显示 */
    text-align: left;           /* 网页正文文本居左显示 */
    border: solid 1px red;      /* 定义边框，方便观察，可以不定义 */
    width: 90%;                 /* 弹性宽度，只有这样才可以实现居中显示效果 */
}
div#page {/* 网页内套的样式 */
    width: 100%;                /* 显示定义 100%宽度，以便与外套同宽 */
    float: left;                /* 浮动显示 */
}
</style>

<div id="wrap">
    <div id="page">网页内套</div>
</div>
```



图 10.13　让浮动页面居中显示

【示例 3】本示例演示定位页面如何实现居中显示。

定位布局相对复杂，要实现居中显示，可以借助内外两个包含框来实现，设计外框为相对定位，内框为绝对定位显示。这样内框将根据外框进行定位，由于外框为相对定位，将遵循流动布局的特征进行布局。页面主要 CSS 样式代码如下。

```
<style type="text/css">
div#wrap{/* 网页外套的样式 */
```






```
        margin-left:auto;                   /* 左侧边界自动显示 */
        margin-right:auto;                  /* 右侧边界自动显示 */
        text-align:left;                    /* 网页正文文本居左显示 */
        border:solid 1px red;               /* 定义边框，方便观察，可以不定义 */
        width:80%;                          /* 液态宽度，只有这样才可以实现居中显示效果 */
        position:relative;                  /* 定义网页外框相对定位，设计包含块 */
}
div#page {/* 网页内套样式 */
 width:100%;                                /* 与外套同宽 */
        position:absolute;                  /* 绝对定位 */
}
</style>
<div id="wrap">
        <div id="page">网页内套</div>
</div>
```

10.3.2　设计 HTML5 文档弹性显示



边界可以设置为百分比，百分比的取值是根据父元素宽度来计算的。使用百分比的好处是能够使页面自适应窗口大小，并能够及时调整边界宽度。从这点考虑，选用百分比具有更大的灵活性和更多的使用技巧。

【示例】继续以上节示例的 HTML5 文档结构为基础，本示例通过 margin 取值百分比定义弹性布局页面，代码如下，效果如图 10.14 所示。

```
<style type="text/css">
body {
        padding: 0;                     /* 清除页边距 */
        margin: 0;                      /* 清除页边距 */
}
div#page {/* 网页样式 */
        margin-left: 5%;                /* 左侧边界 */
        margin-right: 5%;               /* 右侧边界 */
        padding:12px;                   /* 增加页边距 */
        background:rgba(0,0,200,.2)     /* 增加网页背景色，以方便观察 */
}
</style>
```











图 10.14　弹性布局效果

在上面的示例中，把所有边界都设置为百分比，这样当窗口发生变化时，显示内容也比较得体地成比例变化，不至于当窗口很小时，段落文本所占区域比例很大，当窗口很大时，段落文本所占区域比例又显小气。边界的随机应变使页面显得更灵活。

注意： 如果父元素的宽度发生变化，则边界宽度也会随之变化，整个版面可能会混乱，因此在综合布局时要慎重选择。不过在结构单纯、内容单一的布局中，适当使用会使页面更具人性化和多变效果。



10.3.3　调整 HTML5 栏目显示顺序

下面利用 margin 取负值的设计技巧来实现栏目位置的调换显示。本例继续以上一节的 HTML5 文档结构为基础进行练习。

【操作步骤】

第 1 步，新建 HTML5 文档，保存为 test1.html，复制上节示例的文档结构。首先，对其结构进行优化，代码如下，优化的目的是方便使用 float 实现多列布局。

```
<div id="page">
    <header>
        <nav>
            <ul>
                <li><a href="#">首页</a></li>
                <li><a href="#">站内新闻</a></li>
                <li><a href="#">站外新闻</a></li>
            </ul>
        </nav>
    </header>
    <main>
        <h1>站内新闻</h1>
```



```
        <nav>
            <ul>
                <li><a href="#">HTML5</a></li>
                <li><a href="#">CSS3</a></li>
                <li><a href="#">JavaScript</a></li>
            </ul>
        </nav>
        <article>
            <h2 id="web">W3C</h2>
            <h3>W3C 中国区会员沙龙在北京航空航天大学举行</h3>
            <p>2017 年 9 月 14 日，W3C 在北京航空航天大学举办了中国区会员沙龙活动，向到会的中
国区会员代表介绍 W3C 目前标准工作进展及计划，并提供一个新老朋友参与 W3C 及其他相关话题问答与互
动讨论的交流平台。</p>
        </article>
        <aside>
            <h2 id="new">最新新闻</h2>
            <ul>
                <li>W3C 发布 ODRL 信息模型、ODRL 词汇表及表达两份候选推荐标准 征集参考实现
及审阅意见</li>
                <li>W3C 技术研讨会：Web 虚拟现实编著—机遇与挑战</li>
                <li>W3C 发布核心无障碍 API 映射（Core-AAM）1.1 版候选推荐标准 征集参考实现</li>
            </ul>
            <h2 id="blog">W3C 官方博客</h2>
            <ul>
                <li>W3C 启动 WebAssembly 工作组</li>
                <li>W3C 数据的未来方向</li>
                <li>W3C 数字出版主要进展</li>
            </ul>
        </aside>
    </main>
    <footer>本站由北京航空航天大学（W3C/Beihang）维护 京 ICP 备 05004617-3 文保网安备案号
1101080018</footer>
</div>
```

在重构过程中，我们没有破坏原有文档结构，仅对<main>结构进行分组，分别使用<article>和<aside>包裹住文章块和侧栏栏目，这样方便使用 CSS 进行控制。

第 2 步，设计页面居中显示，并定义页面宽度为 960 像素，为下面操作提供方便，代码如下。




Note

```
body {
    padding: 0;             /* 清除页边距 */
    margin: 0;              /* 清除页边距 */
    text-align: center;     /* 网页居中显示（IE 浏览器有效） */
}
div#page {/* 网页外套的样式 */
    margin-left: auto;      /* 左侧边界自动显示 */
    margin-right: auto;     /* 右侧边界自动显示 */
    text-align: left;       /* 恢复网页文本居左显示 */
    width: 960px;           /* 固定宽度，只有这样才可以实现居中显示效果 */
}
```

第 3 步，设计标题栏样式。先清除列表结构的默认样式；然后设计菜单项浮动显示，设计导航条效果；最后使用伪对象选择器在标题栏容器尾部增加一个空对象，让其以块显示，并清除左右两侧浮动，这样能够强迫标题栏撑开显示，代码如下。

```
header nav ul {/* 清除默认样式 */
    padding: 0; margin: 0;
    list-style-type: none;
}
header nav ul li {/* 向左浮动显示 */
    float: left;
    width: 100px; height: 40px;         /* 固定高度和宽度 */
    line-height: 40px;                  /* 垂直居中显示 */
    text-align: center;                 /* 水平居中显示 */
}
header nav:after { /* 撑开标题栏 */
    content: "";                        /* 添加空对象 */
    display: block;                     /* 块显示 */
    clear: both;                        /* 清除左右浮动 */
}
```

第 4 步，设计<nav>、<article>、<aside>3 个栏目向左浮动显示，并固定宽度；然后隐藏不需要的标题文本；最后在<main>尾部添加一个空对象，定义块显示，清除左右浮动，目的是为了撑开<main>容器，代码如下。

```
main nav, main article, main aside { float: left; }      /* 定义三栏浮动显示 */
/* 定义三栏固定宽度 */
main nav { width: 200px; }
main article { width: 460px; }
```






```
main aside { width: 300px; }
main h1 { display: none; }
article h2 { display: none; }
/* 撑开 main 容器 */
main:after {
    content: "";
    display: block;
    clear: both;
}
```

第 5 步，设计脚注区块样式，并为标题栏和脚注栏增加背景色，以区分各行栏目，代码如下。此时，页面显示效果如图 10.15 所示。

```
footer {
    padding: 12px;
    text-align: center;
    font-size: 12px;
}
header { background: rgba(0,0,255,0.2); }
footer { background: rgba(255,0,0,0.2); }
```

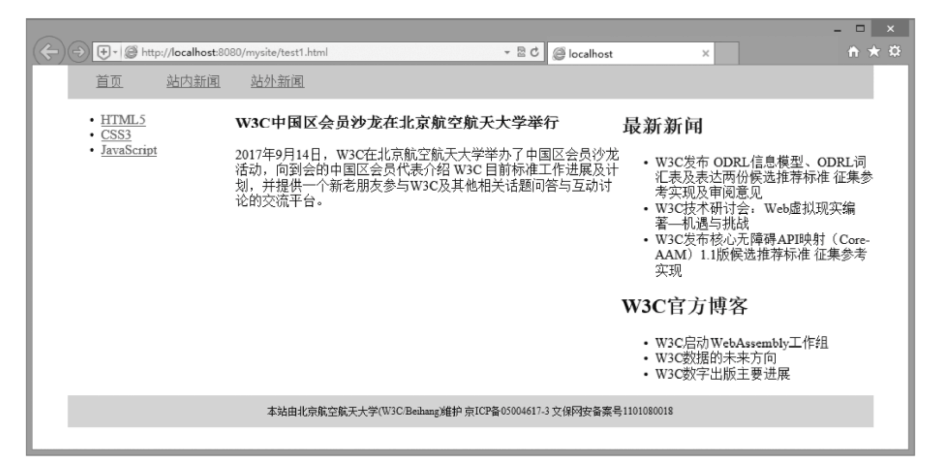


图 10.15　初步完成页面布局

第 6 步，把 test1.html 另存为 test2.html，然后在 test2.html 文档的内部样式表中添加如下样式。注意，要正确计算 margin 的设置值。

```
main nav { margin-left:760px;}          /* article 的宽度+aside 的宽度 */
main article { margin-left:-660px;}     /* article 的宽度+nav 的宽度 */
main aside { margin-left:-960px;}       /* article 的宽度+aside 的宽度+nav 的宽度*/
```





通过 CSS 的 margin 调整<nav>、<article>、<aside>3 个栏目的显示顺序，则演示效果如图 10.16 所示。

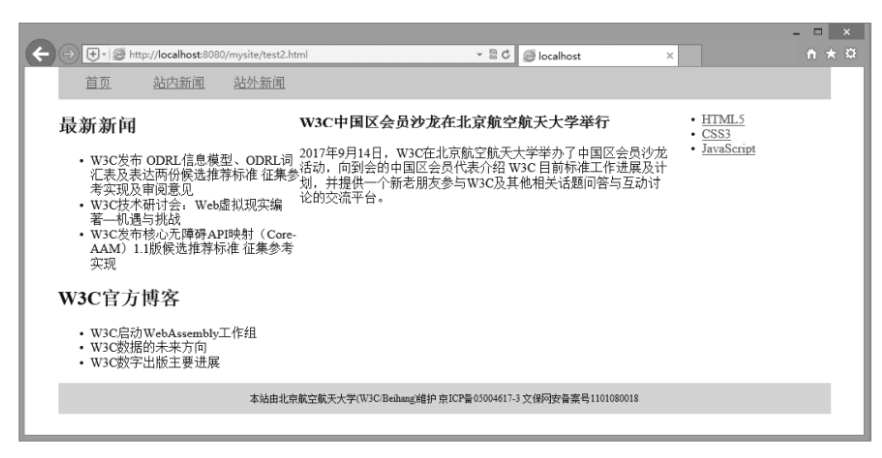


图 10.16　margin 取负值时的布局效果


视 频 讲 解

10.3.4　禁止 HTML5 栏目错行显示

在复杂的页面设计中，用户可能需要在设计好的页面中添加边框、补白、边界等样式，此时就会破坏整个页面结构。

【示例】以上节示例为例进行说明，我们需要为中间 3 栏添加 padding，以避免栏目内容挤在一起。同时为了看清每个栏目的边界，再为每个栏目添加一个边框线，代码如下。

```
main nav, main article, main aside {
    border: solid 1px red;          /* 增加边框线，方便观察 */
    padding: 12px;                  /* 增加栏目之间空隙 */
}
```

在浏览器中预览，显示效果如图 10.17 所示。

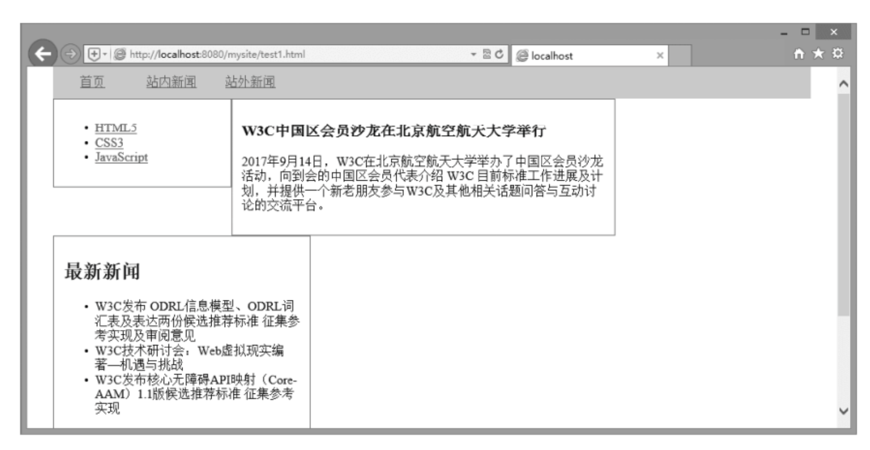


图 10.17　浮动错行问题



分析原因：在修改布局前后，没有考虑到 CSS 盒模型对布局的影响。

如果不怕麻烦，可以调整 CSS 样式代码，修改每行每列的宽度，确保每行总宽度不大于包含框的宽度。

把 test1.html 另存为 test2.html，然后在 test2.html 文档的内部样式表中添加如下样式，为每个栏目框定义 box-sizing: border-box;。

```
main nav, main article, main aside {
    border: solid 1px red;
    padding: 12px;
    box-sizing: border-box;  /*  以怪异模式解析，这样 width 可以包含 border 和 padding */
}
```

重新在浏览器中预览，则布局效果如图 10.18 所示。关于 box-sizing 属性的说明可以参考 10.2.1 节内容。

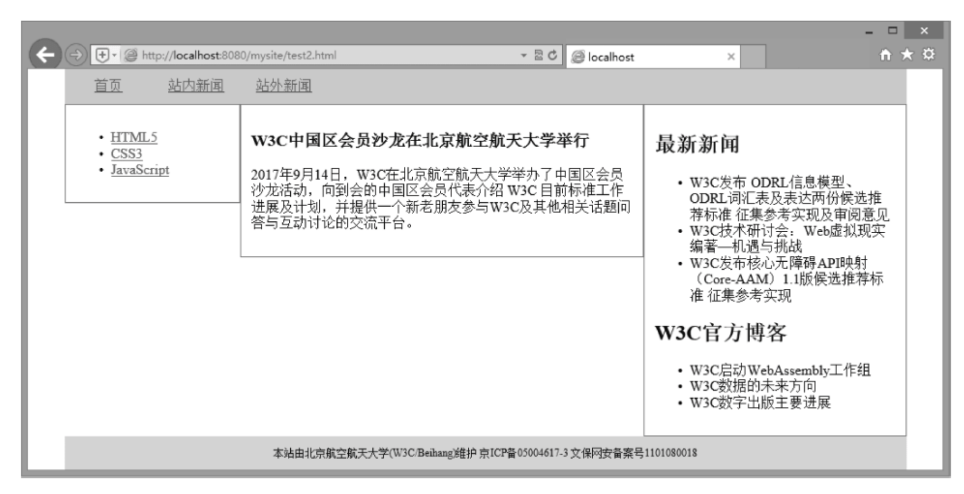


图 10.18　改善浮动布局错行问题

10.3.5　设计 HTML5 多栏等高显示



在多栏布局中，由于每个栏目内容高度不一致，会出现栏目高度参差不齐的现象。

【示例 1】 以上节示例为例进行说明。

第 1 步，把 test2.html 另存为 test1.html，然后在 test1.html 文档的内部样式表中添加如下样式，为<nav>、<article>、<aside>3 个栏目定义不同的背景色，同时清除 border: solid 1px red;声明。

```
main nav { background: rgba(0,255,255,0.2); }
main article { background: rgba(255,255,0,0.2); }
main aside { background: rgba(255,0,255,0.2); }
```

第 2 步，在浏览器中预览，效果如图 10.19 所示。







图 10.19　栏目高度不一致

为了解决 3 栏高度不齐的问题，下面介绍两种方法。

方法一：使用伪列布局。所谓伪列布局，就是设计一个背景图像，利用背景图像来模拟栏目的背景（方法二见示例 2）。

第 3 步，以上节示例结构为基础，使用 Photoshop 设计一个长条形的背景图，长度与页面宽度保持一致，高度任意，如图 10.20 所示。

图 10.20　设计伪列布局背景图像

第 4 步，为<main>包含框定义背景图，让其沿 y 轴平铺，代码如下。

```
main{
    background:url(images/bg.png) repeat-y;        /* 伪列背景图像，y 轴平铺 */
    float:left;                                    /* 浮动显示，以便撑开 main 容器，显示背景 */
    width:100%;                                    /* 满页显示 */
}
```

第 5 步，为了避免栏目背景色的干扰，不妨在 CSS 样式表中删除各个栏目背景色声明，同时删除 main:after 伪对象样式，该对象会影响浮动布局背景色的垂直平铺，代码如下。

```
main:after {/* 为了撑开 main 框，临时添加伪对象 */
    content: "";
    display: block;
    clear: both;
}
main nav { background: rgba(0,255,255,0.2); }
main article { background: rgba(255,255,0,0.2); }
main aside { background: rgba(255,0,255,0.2); }
```

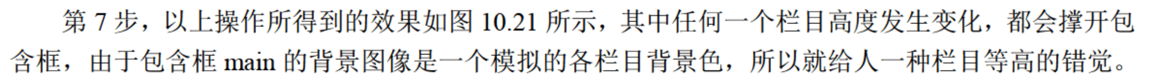




第 6 步，为 footer 添加 clear:both;声明，防止页脚内容向上环绕显示，代码如下。

```
footer {
    clear:both;
}
```

第 7 步，以上操作所得到的效果如图 10.21 所示，其中任何一个栏目高度发生变化，都会撑开包含框，由于包含框 main 的背景图像是一个模拟的各栏目背景色，所以就给人一种栏目等高的错觉。



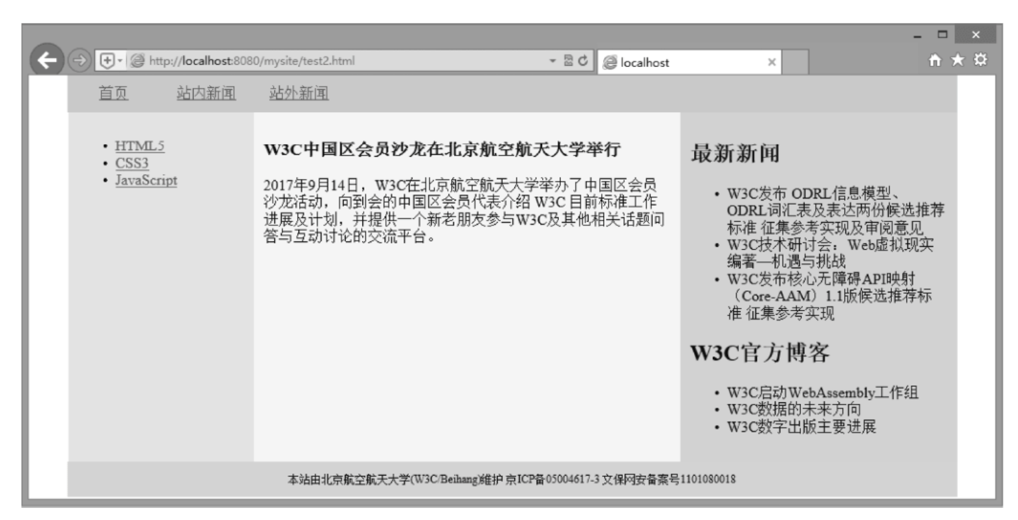


图 10.21　伪列布局效果

> **提示：** 在使用这种方法时，一定确保页面宽度是固定的，不能够设计为弹性页面（百分比宽度），或者宽度值为 auto。

【示例 2】 示例 1 使用固定宽度的背景图像设计伪列布局，这种方法存在局限性，本例使用 CSS3 渐变背景来改善这种设计方法，使其能够适应弹性页面显示。例如，针对示例 1，修改 main 包含框的 background 属性值，使用 linear-gradient()函数定义直线渐变背景图像，从左到右为其设计 3 个色块：左侧 200 像素背景色为 rgba(0,255,255,0.2)，中间色块从 200 像素到 660 像素，背景色为 rgba(255,255,0,0.2)，右侧色块从 660 像素开始，背景色为 rgba(255,0,255,0.2)，代码如下。

```
main{
    background: linear-gradient(to right, rgba(0,255,255,0.2) 200px, rgba(255,255,0,0.2) 0, rgba(255,255,0,0.2)
660px,  rgba(255,0,255,0.2) 0) ;
    float:left;
    width:100%;
}
```

在浏览器中预览，效果如图 10.22 所示。这样我们可以通过设计渐变背景的色标位置为百分比值，来适应弹性布局的页面。





Note



图 10.22　伪列布局效果

【拓展】

另一种解决 3 栏高度不齐问题的方法是使用补白和边界重叠法。这种方法的思路是：设计<nav>、<article>、<aside>3 个栏目的底部补白为无穷大，这样在有限的窗口内都能够显示栏目的背景色，因此也就不用担心栏目高度无法自适应。为了避免补白过大产生的空白区域，再设计底部边界为负无穷大，从而覆盖掉多出来的补白区域，最后在中间行包含框中定义 overflow:hidden;声明剪切掉多出的区域即可。

把 test3.html 另存为 test4.html，将下面样式代码放置到 test4.html 示例中，并删除伪列布局中为<main>标签定义的背景图像，代码如下，此时可以得到如图 10.23 所示的效果。

```
main { overflow:hidden; }/* 剪切多出的区域 */
main nav{
    padding-bottom:9999px;                    /* 定义底部补白无穷大 */
    margin-bottom:-9999px;                    /* 定义底部边界负无穷大 */
}
main article {
    padding-bottom:9999px;                    /* 定义底部补白无穷大 */
    margin-bottom:-9999px;                    /* 定义底部边界负无穷大 */
}
main aside {
    padding-bottom:9999px;                    /* 定义底部补白无穷大 */
    margin-bottom:-9999px;                    /* 定义底部边界负无穷大 */
}
main nav { background: rgba(0,255,255,0.2); }
main article { background: rgba(255,255,0,0.2); }
main aside { background: rgba(255,0,255,0.2); }
```







图 10.23　使用补白和边界重叠法设计自适应高度的布局

10.4　在 线 练 习

1. 使用 HTML 结构标签设计各种网页模块。
2. 通过练习了解 CSS3 布局新特性。







在 线 练 习 2



第11章 使用CSS3设计弹性布局

2009年，W3C提出一种崭新的布局方案——伸缩盒布局，它可以简便、完整、响应式地实现各种页面布局，自由设置多个栏目在一个容器中的分布方式，以及合理处理容器内可用的空间。使用该模型可以轻松创建自适应窗口的流动布局或者自适应字体大小的弹性布局。W3C的伸缩盒布局分为旧版本、新版本，以及混合过渡版本3种不同的编码方式。其中混合过渡版本主要是针对IE10做了兼容。目前该技术多应用在移动端网页布局，本章将主要讲解旧版本和新版本伸缩盒布局的基本用法。

权威参考：http://www.w3.org/TR/css-flexbox-1/。

【学习要点】

- 设计多列布局。
- 设计旧版伸缩盒布局。
- 设计新版伸缩盒布局。









11.1　多 列 布 局

CSS3 新增了 columns 属性，用来设计多列布局，它允许网页内容跨栏显示，适合设计正文版式。

权威参考：http://www.w3.org/TR/css3-multicol/。

11.1.1　设置列宽

column-width 属性可以定义单列显示的宽度，基本语法如下。

```
column-width: <length> | auto
```

取值简单说明如下。

☑　<length>：用长度值来定义列宽。不允许有负值。

☑　auto：根据<'column-count'>自定分配宽度，为默认值。

【示例】本示例演示了 column-width 属性在多列布局中的应用。设计 body 元素的列宽度为 300 像素，如果网页内容能够在单列内显示，则会以单列显示；如果窗口足够宽，且内容很多，则会在多列中显示，效果如图 11.1 所示，根据窗口宽度自动调整为两栏显示，列宽度显示为 300 像素，代码如下。

```
<style type="text/css">
/*定义网页列宽为 300 像素，则网页中每个栏目的最大宽度为 300 像素*/
body {column-width:300px;}
h1 {color: #333333; padding: 5px 8px;font-size: 20px;text-align: center; padding: 12px;}
h2 {font-size: 16px; text-align: center;}
p {color: #333333; font-size: 14px; line-height: 180%; text-indent: 2em;}
</style>

<h1>W3C 标准</h1>
<p>W3C 的各类技术标准在努力为各类应用的开发打造一个<strong>开放的 Web 平台（Open Web Platform）</strong>。尽管这个开放 Web 平台的边界在不断延伸，产业界认为 HTML5 将是这个平台的核心，平台的能力将依赖于 W3C 及其合作伙伴正在创建的一系列 Web 技术，包括 CSS，SVG，WOFF，语义 Web，及 XML 和各类应用编程接口（APIs）。</p>
<p>截至 2014 年 3 月，W3C 共设立 5 个技术领域，开展 23 个标准计划。W3C 设有 46 个工作组（Working Group）、14 个兴趣小组（Interest Group）、3 个协调组（Coordination Group）、169 个社区组（Community Group），以及 3 个业务组（Business Group）。</p>
<p>目前，W3C 正在探讨技术专家及个人参与 W3C 标准制定过程的 Webizen 计划，敬请期待。</p>
<p>W3C 于 2014 年 11 月发布了题为&ldquo;W3C 工作重点（2014 年 11 月）”的报告，这是最新的一份对 W3C 近期开展的工作要点进行了综述的文章，阐述了近期的工作重点和优先级。</p>
```







图 11.1　固定列表宽度显示



11.1.2　设置列数

column-count 属性可以定义显示的列数，基本语法如下。

```
column-count：<integer> | auto
```

取值简单说明如下。

☑　<integer>：用整数值来定义列数。不允许有负值。

☑　auto：根据<' column-width '>自定义分配宽度，为默认值。

【示例】 在 11.1.1 节示例的基础上，如果定义网页列数为 3，则不管浏览器窗口怎么调整，页面内容总是遵循 3 列布局。代码如下，演示效果如图 11.2 所示。

```
/*定义网页列数为 3，这样整个页面总是显示为 3 列*/
body { column-count:3;}
```

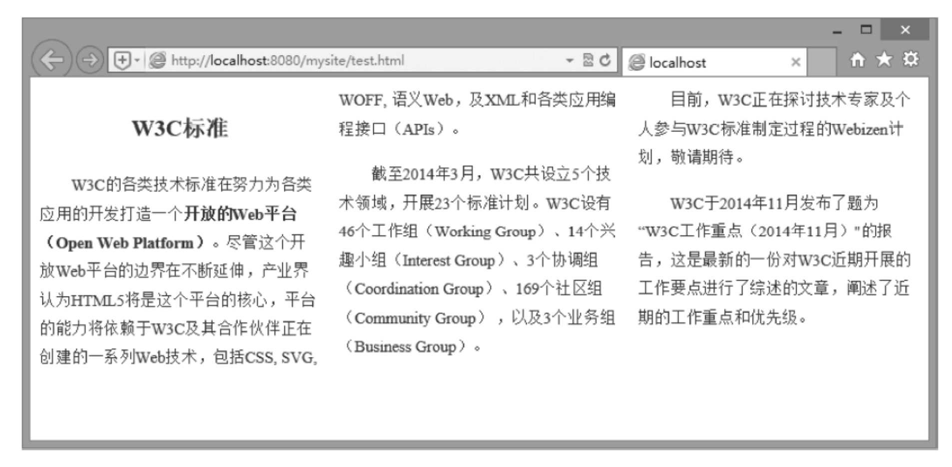


图 11.2　设计 3 列显示







11.1.3　设置间距

column-gap 属性可以定义两栏之间的间距，基本语法如下。

```
column-gap：<length> | normal
```

取值简单说明如下。

☑　<length>：用长度值来定义列与列之间的间隙。不允许有负值。

☑　normal：　与<'font-size'>大小相同。假设该对象的 font-size 为 16px，则 normal 值为 16px，依此类推。

【示例】在 11.1.2 节示例的基础上，通过 column-gap 和 line-height 属性配合使用，把文档版面设计得疏朗大方，以方便阅读。其中列间距为 3em，行高为 2.5em，页面内文字内容看起来更明晰。代码如下，效果如图 11.3 所示。

```
body {
    /*定义页面内容显示为 3 列*/
    column-count: 3;
    /*定义列间距为 3em，默认为 1em*/
    column-gap: 3em;
    line-height: 2.5em; /*  定义页面文本行高  */
}
```

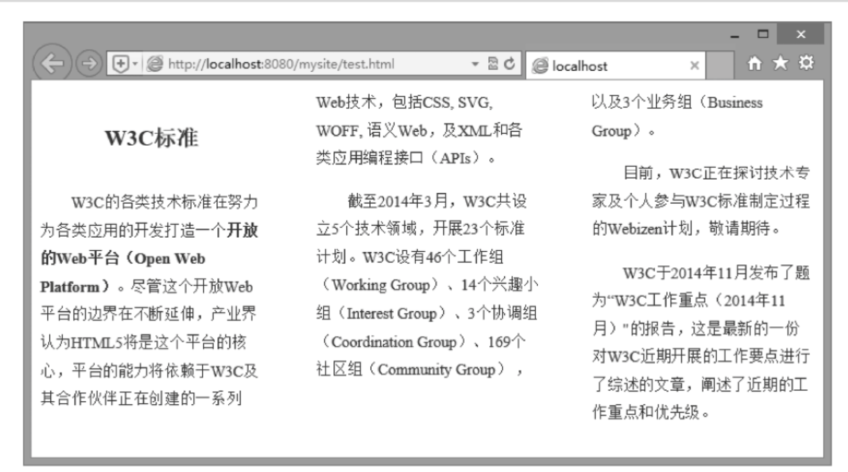


图 11.3　设计疏朗的跨栏布局

11.1.4　设置列边框



column-rule 属性可以定义每列之间边框的宽度、样式和颜色。基本语法如下。

```
column-rule：<' column-rule-width '> || <' column-rule-style '> || <' column-rule-color '>
```

取值简单说明如下。



```
<' column-rule-width '>：设置对象的列与列之间的边框厚度。
<' column-rule-style '>：设置对象的列与列之间的边框样式。
<' column-rule-color '>：设置对象的列与列之间的边框颜色。
```



Note

column-rule-style 属性语法如下，取值与边框样式 border-style 相同。

```
column-rule-style：none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset
```

column-rule-width 与 border-width，column-rule-color 与 border-color 设置相同，在此就不再重复说明。

【示例】 在 11.1.3 节示例的基础上，为每列之间的边框定义一个虚线分割线，线宽为 2 像素，灰色显示。代码如下，演示效果如图 11.4 所示。

```
body {
    /*定义页面内容显示为 3 列*/
    column-count: 3;
    /*定义列间距为 3em，默认为 1em*/
    column-gap: 3em;
    line-height: 2.5em;
    /*定义列边框为 2 像素宽的灰色虚线*/
    column-rule: dashed 2px gray;
}
```

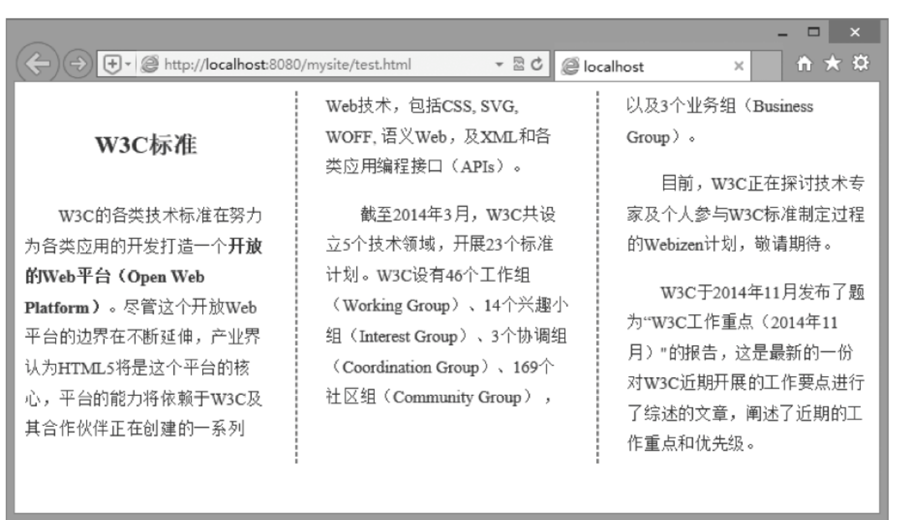


图 11.4　设计列边框效果



视频讲解

11.1.5　设置跨列显示

column-span 属性可以定义跨列显示，基本语法如下。

```
column-span：none | all
```





取值简单说明如下。

☑　none：不跨列。

☑　all：横跨所有列。

【示例】在 11.1.4 节示例的基础上，使用 column-span 属性定义一级标题跨列显示。代码如下，演示效果如图 11.5 所示。

```
body {
    /*定义页面内容显示为 3 列*/
    column-count: 3;
    /*定义列间距为 3em，默认为 1em*/
    column-gap: 3em;
    line-height: 2.5em;
    /*定义列边框为 2 像素宽的灰色虚线*/
    column-rule: dashed 2px gray;
    }
    /*设置一级标题跨越所有列显示*/
h1 {
    color: #333333; font-size: 20px; text-align: center;
    padding: 12px;
    /*跨越所有列显示*/
    column-span: all;
}
p {color: #333333; font-size: 14px; line-height: 180%; text-indent: 2em;}
```

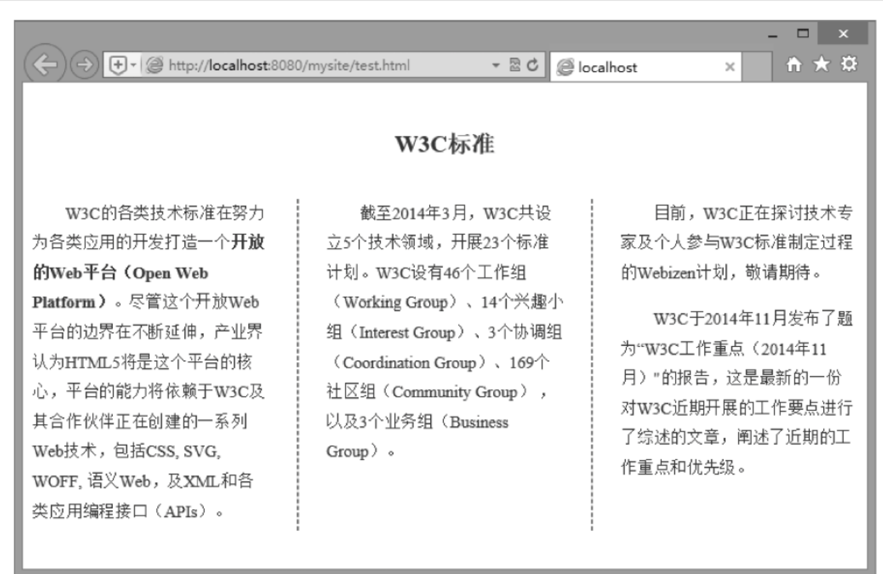


图 11.5　设计标题跨列显示效果







11.1.6　设置列高度

column-fill 属性可以定义栏目的高度是否统一，基本语法如下。

```
column-fill：auto | balance
```

取值简单说明如下。

☑　auto：列高度自适应内容。

☑　balance：所有列的高度以其中最高的一列为准。

【示例】在上面示例的基础上，使用 column-fill 属性定义每列高度一致，代码如下。

```
body {
    /*定义页面内容显示为 3 列*/
    column-count: 3;
    /*定义列间距为 3em，默认为 1em*/
    column-gap: 3em;
    line-height: 2.5em;
    /*定义列边框为 2 像素宽的灰色虚线*/
    column-rule: dashed 2px gray;
    /*设置各列高度一致*/
    column-fill: balance;
}
```

11.2　旧版伸缩盒

Flexbox（伸缩盒）是 CSS3 新增的布局模型，实际上它一直都存在。最开始它作为 Mozilla XUL 的一个功能被用来制作程序界面，如 Firefox 的工具栏就多次使用这个属性。本节重点介绍老版本伸缩盒模型的基本用法。

11.2.1　启动伸缩盒

在旧版本中启动伸缩盒模型，只需设置容器的 display 的属性值为 box（或 inline-box），用法如下。

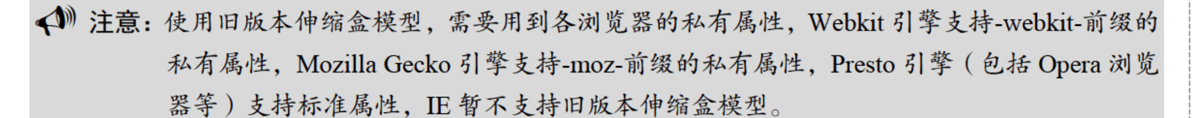
```
display: box;
display: inline-box;
```

伸缩盒模型由父容器和子容器两部分构成。父容器通过 display:box;或者 display: inline-box;启动伸缩盒布局功能。子容器通过 box-flex 属性定义布局宽度，以及如何对父容器的宽度进行分配。父容器又通过如下属性定义包含容器的显示属性，简单说明如下。





- ☑ box-orient：定义父容器里子容器的排列方式是水平还是垂直。
- ☑ box-direction：定义父容器里的子容器排列顺序。
- ☑ box-align：定义子容器的垂直对齐方式。
- ☑ box-pack：定义子容器的水平对齐方式。



注意：使用旧版本伸缩盒模型，需要用到各浏览器的私有属性，Webkit 引擎支持-webkit-前缀的私有属性，Mozilla Gecko 引擎支持-moz-前缀的私有属性，Presto 引擎（包括 Opera 浏览器等）支持标准属性，IE 暂不支持旧版本伸缩盒模型。

11.2.2　设置宽度



在默认情况下，盒子没有弹性，它将尽可能宽地使其内容可见，且没有溢出，其大小由 width、height、min-height、min-width、max-width 或者 max-height 属性值来决定。

使用 box-flex 属性可以把默认布局变为盒布局。如果 box-flex 的属性值为 1，则元素变得富有弹性，其大小将按以下方式计算。

- ☑ 声明的大小（width、height、min-width、min-height、max-width、max-height）。
- ☑ 父容器的大小和所有余下的可利用的内部空间。

如果盒子没有声明大小，那么其大小将完全取决于父容器的大小，即盒子的大小等于父容器的大小乘以其 box-flex 在所有盒子 box-flex 总和中的百分比，用公式表示如下。

盒子的大小 = 父容器的大小 × 盒子的 box-flex / 所有盒子的 box-flex 值的和

余下的盒子将按照上面的原则分享剩下的可用空间。

【示例】本示例定义左侧边栏的宽度为 240px，右侧边栏的宽度为 200px，中间内容版块的宽度将由 box-flex 属性确定。详细代码如下，演示效果如图 11.6 所示。当调整窗口宽度时，中间列的宽度会自适应显示，使整个页面总是满窗口显示。

```
<style type="text/css">
#container {
    /*定义弹性盒布局样式*/
    display: -moz-box;
    display: -webkit-box;
    display: box;
}
#left-sidebar {
    width: 240px;
    padding: 20px;
    background-color: orange;
}
```




```
#contents {
    /*定义中间列宽度为自适应显示*/
    -moz-box-flex: 1;
    -webkit-box-flex: 1;
    flex: 1;
    padding: 20px;
    background-color: yellow;
}
#right-sidebar {
    width: 200px;
    padding: 20px;
    background-color: limegreen;
}
#left-sidebar, #contents, #right-sidebar {
    /*定义盒样式*/
    -moz-box-sizing: border-box;
    -webkit-box-sizing: border-box;
    box-sizing: border-box;
}
</style>

<div id="container">
    <div id="left-sidebar">
        <h2>宋词精选</h2>
        <ul>
            <li><a href="">卜算子·咏梅</a></li>
            <li><a href=""> 声声慢·寻寻觅觅</a></li>
            <li><a href=""> 雨霖铃·寒蝉凄切</a></li>
            <li><a href="">卜算子·咏梅</a></li>
            <li><a href="">更多</a></li>
        </ul>
    </div>
    <div id="contents">
        <h1>水调歌头·明月几时有</h1>
        <h2>苏轼</h2>
        <p>丙辰中秋，欢饮达旦，大醉，作此篇，兼怀子由。</p>
        <p>明月几时有？把酒问青天。不知天上宫阙，今夕是何年。我欲乘风归去，又恐琼楼玉宇，高
处不胜寒。起舞弄清影，何似在人间？</p>
```



```
            <p>转朱阁，低绮户，照无眠。不应有恨，何事长向别时圆？人有悲欢离合，月有阴晴圆缺，此
事古难全。但愿人长久，千里共婵娟。</p>
        </div>
        <div id="right-sidebar">
            <h2>词人列表</h2>
            <ul>
                <li><a href="">陆游</a></li>
                <li><a href="">李清照</a></li>
                <li><a href="">苏轼</a></li>
                <li><a href="">柳永</a></li>
            </ul>
        </div>
    </div>
```

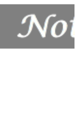




图 11.6　定义自适应宽度

11.2.3　设置顺序



使用 box-ordinal-group 属性可以改变子元素的显示顺序。语法格式如下。

```
box-ordinal-group: <integer>
```

<integer>用整数值来定义伸缩盒对象的子元素显示顺序，默认值为 1。浏览器在显示时，将根据该值从小到大来显示这些元素。

【示例】 以上节示例为基础，在左栏、中栏、右栏中分别加入一个 box-ordinal-group 属性，并指定显示的序号，这里将中栏设置为 1，右栏设置为 2，左栏设置为 3，则可以发现 3 栏显示顺序发生了变化。代码如下，演示效果如图 11.7 所示。





Note

```
#left-sidebar {
    -moz-box-ordinal-group: 3;
    -webkit-box-ordinal-group: 3;
    box-ordinal-group: 3;
}
#contents {
    -moz-box-ordinal-group: 1;
    -webkit-box-ordinal-group: 1;
    box-ordinal-group: 1;
}
#right-sidebar {
    -moz-box-ordinal-group: 2;
    -webkit-box-ordinal-group: 2;
    box-ordinal-group: 2;
}
```



图 11.7　定义列显示顺序



视频讲解

11.2.4　设置方向

使用 box-orient 属性可以定义元素的排列方向，语法格式如下。

```
box-orient: horizontal | vertical | inline-axis | block-axis
```

取值简单说明如下。

☑　horizontal：设置伸缩盒对象的子元素从左到右水平排列。

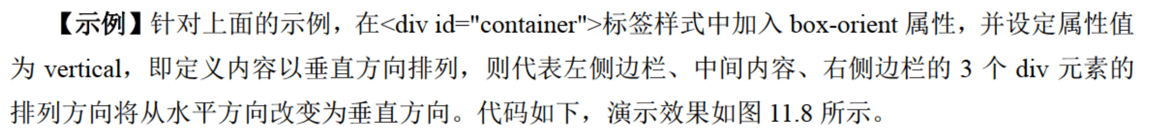




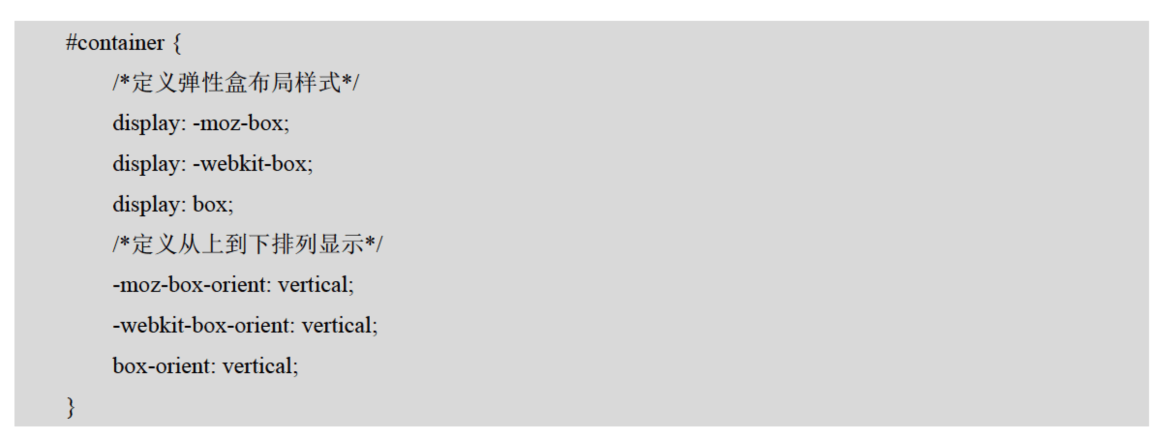
☑　vertical：设置伸缩盒对象的子元素从上到下纵向排列。

☑　inline-axis：设置伸缩盒对象的子元素沿行轴排列。

☑　block-axis：设置伸缩盒对象的子元素沿块轴排列。



【示例】针对上面的示例，在<div id="container">标签样式中加入 box-orient 属性，并设定属性值为 vertical，即定义内容以垂直方向排列，则代表左侧边栏、中间内容、右侧边栏的 3 个 div 元素的排列方向将从水平方向改变为垂直方向。代码如下，演示效果如图 11.8 所示。



```
#container {
    /*定义弹性盒布局样式*/
    display: -moz-box;
    display: -webkit-box;
    display: box;
    /*定义从上到下排列显示*/
    -moz-box-orient: vertical;
    -webkit-box-orient: vertical;
    box-orient: vertical;
}
```



图 11.8　定义列显示方向

使用 box-direction 属性可以让各个子元素反向排序，语法格式如下。

```
box-direction：normal | reverse
```



取值简单说明如下。

☑ normal：设置伸缩盒对象的子元素按正常顺序排列。

☑ reverse：反转伸缩盒对象的子元素的排列顺序。

Note

视频讲解

11.2.5 设置对齐方式

使用 box-pack 属性可以设置子元素水平方向对齐方式，语法格式如下。

```
box-pack：start | center | end | justify
```

取值简单说明如下。

☑ start：设置伸缩盒对象的子元素从开始位置对齐，为默认值。

☑ center：设置伸缩盒对象的子元素居中对齐。

☑ end：设置伸缩盒对象的子元素从结束位置对齐。

☑ justify：设置伸缩盒对象的子元素两端对齐。

使用 box-align 属性可以设置子元素垂直方向对齐方式，语法格式如下。

```
box-align：start | end | center | baseline | stretch
```

取值简单说明如下。

☑ start：设置伸缩盒对象的子元素从开始位置对齐。

☑ center：设置伸缩盒对象的子元素居中对齐。

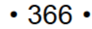
☑ end：设置伸缩盒对象的子元素从结束位置对齐。

☑ baseline：设置伸缩盒对象的子元素基线对齐。

☑ stretch：设置伸缩盒对象的子元素自适应父元素尺寸。

【示例】在本示例中有一个<div class="login">容器，其中包含一个登录表单对象，为了方便练习，本例使用一个<img>标签模拟，然后使用 box-pack 和 box-align 属性让表单对象在<div class="login">容器的正中央显示。同时，设计<div class="login">容器高度和宽度都为 100%，这样就可以让表单对象在窗口中央位置显示。具体实现代码如下，设计效果如图 11.9 所示。

```
<style type="text/css">
/*清除页边距*/
body { margin: 0; padding: 0;}
div { position: absolute; }
.bg {/*设计遮罩层*/
    width: 100%; height: 100%;
    background: #000; opacity: 0.7;
}
.login {
    /*满屏显示*/
    width:100%; height:100%;
```







```
        /*定义弹性盒布局样式*/
        display: -moz-box;
        display: -webkit-box;
        display: box;
        /*垂直居中显示*/
        -moz-box-align: center;
        -webkit-box-align: center;
        box-align: center;
        /*水平居中显示*/
        -moz-box-pack: center;
        -webkit-box-pack: center;
        box-pack: center;
    }
    </style>

    <div class="web"><img src="images/bg.png" /></div>
    <div class="bg"></div>
    <div class="login"><img src="images/login.png"    /></div>
```

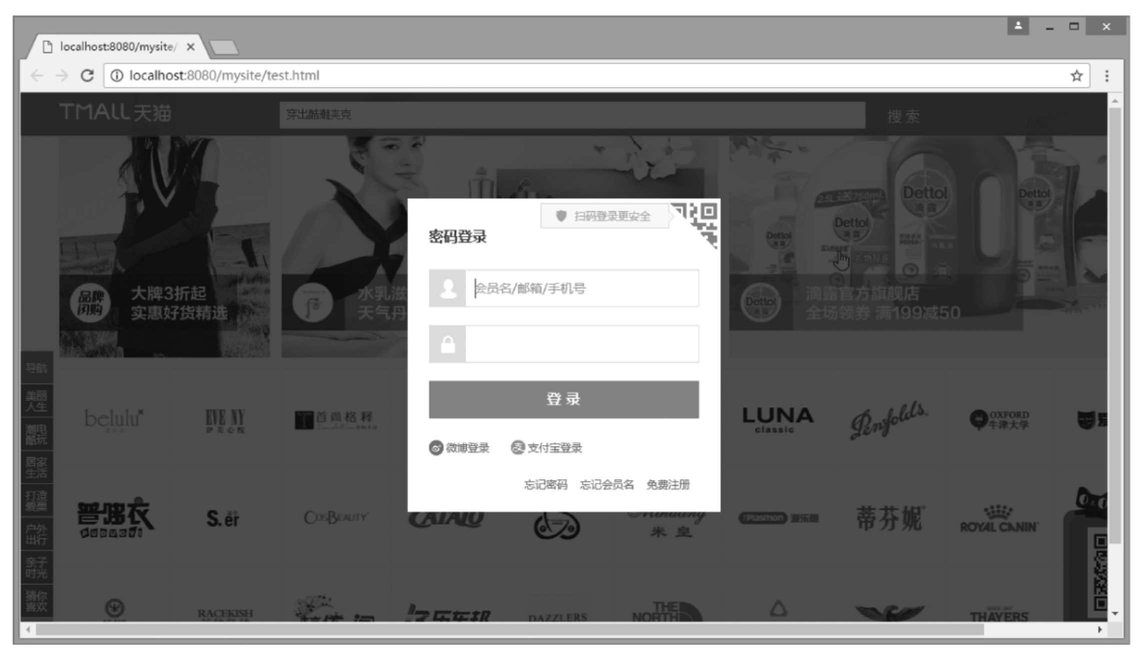


图 11.9　设计登录表单中央显示





11.3　新版伸缩盒

伸缩盒模型是一个新的盒子模型，它主要优化了 UI 布局，可以简单地使一个元素居中（包括水平和垂直居中），可以扩大或收缩元素来填充容器的可利用空间，也可以改变布局顺序等。本节将重点介绍新版本伸缩盒模型的基本用法。

11.3.1　认识 Flexbox 系统

Flexbox 由伸缩容器和伸缩项目组成。

在伸缩容器中，每一个子元素都是一个伸缩项目，伸缩项目可以是任意数量的，伸缩容器外和伸缩项目内的一切元素都不受影响。

伸缩项目沿着伸缩容器内的一个伸缩行定位，通常每个伸缩容器只有一个伸缩行。在默认情况下，伸缩行和文本方向一致：从左至右，从上到下。

常规布局是基于块和文本流方向，而 Flex 布局是基于 flex-flow 流。如图 11.10 所示是 W3C 规范对 Flex 布局的解释。

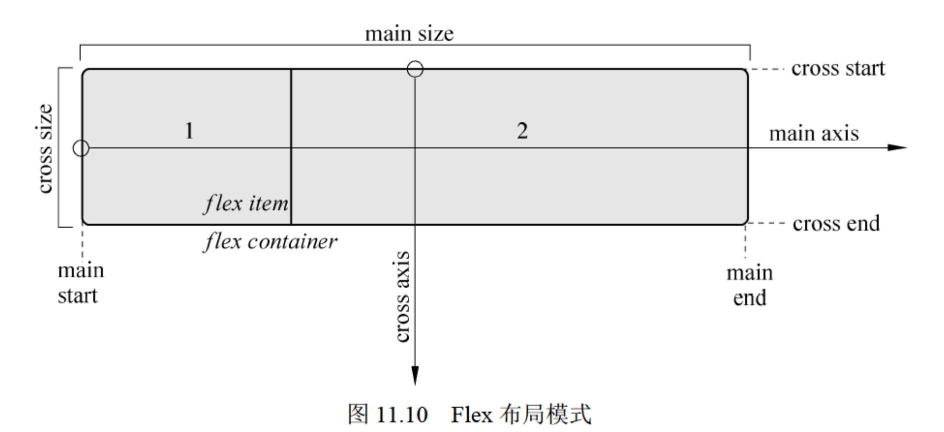


图 11.10　Flex 布局模式

伸缩项目是沿着主轴，从主轴起点到主轴终点，或者沿着侧轴，从侧轴起点到侧轴终点排列。

☑　主轴（main axis）：伸缩容器的主轴，伸缩项目主要沿着这条轴进行排列布局。注意，它不一定是水平的，这主要取决于 justify-content 属性设置。

☑　主轴起点（main-start）和主轴终点（main-end）：伸缩项目放置在伸缩容器内从主轴起点向主轴终点方向。

☑　主轴尺寸（main size）：伸缩项目在主轴方向的宽度或高度就是主轴的尺寸。伸缩项目主要的大小属性是宽度或高度，由对着主轴方向的那个来决定。

☑　侧轴（cross axis）：垂直于主轴，它的方向主要取决于主轴方向。

☑　侧轴起点（cross-start）和侧轴终点（cross-end）：伸缩行的配置从容器的侧轴起点边开始，往侧轴终点边结束。









☑ 侧轴尺寸（cross size）：伸缩项目在侧轴方向的宽度或高度就是项目的侧轴长度，伸缩项目的侧轴长度属性是 width 或 height，由对着侧轴方向的那个来决定。

一个伸缩项目就是一个伸缩容器的子元素，伸缩容器中的文本也被视为一个伸缩项目。伸缩项目中的内容与普通文本流一样。例如，当一个伸缩项目被设置为浮动时，用户依然可以在这个伸缩项目中放置一个浮动元素。

11.3.2　启动伸缩盒

通过设置元素的 display 属性为 flex 或 inline-flex 可以定义一个伸缩容器。设置为 flex 的容器被渲染为一个块级元素，而设置为 inline-flex 的容器则被渲染为一个行内元素。具体语法如下。

```
display: flex | inline-flex;
```

上面的属性值决定容器是行内显示还是块显示，它的所有子元素将变成 flex 文档流，被称为伸缩项目。

此时，CSS 的 columns 属性在伸缩容器上没有效果，同时 float、clear 和 vertical-align 属性在伸缩项目上也没有效果。

【示例】本示例将设计一个伸缩容器，其中包含 4 个伸缩项目。代码如下，演示效果如图 11.11 所示。

```
<style type="text/css">
.flex-container {
    display: -webkit-flex;
    display: flex;
    width: 500px; height: 300px;
    border: solid 1px red;
}
.flex-item {
    background-color: blue;
    width: 200px; height: 200px;
    margin: 10px;
}
</style>

<div class="flex-container">
    <div class="flex-item">伸缩项目 1</div>
    <div class="flex-item">伸缩项目 2</div>
    <div class="flex-item">伸缩项目 3</div>
    <div class="flex-item">伸缩项目 4</div>
</div>
```





Note

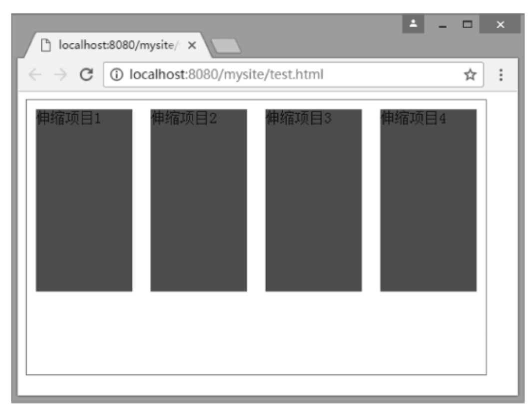


图 11.11　定义伸缩盒布局



视 频 讲 解

11.3.3　设置主轴方向

使用 flex-direction 属性可以定义主轴方向，它适用于伸缩容器。具体语法如下。

```
flex-direction：row | row-reverse | column | column-reverse
```

取值说明如下。

☑　row：主轴与行内轴方向作为默认的书写模式，即横向从左到右排列（左对齐）。

☑　row-reverse：对齐方式与 row 相反。

☑　column：主轴与侧轴方向作为默认的书写模式，即纵向从上往下排列（顶对齐）。

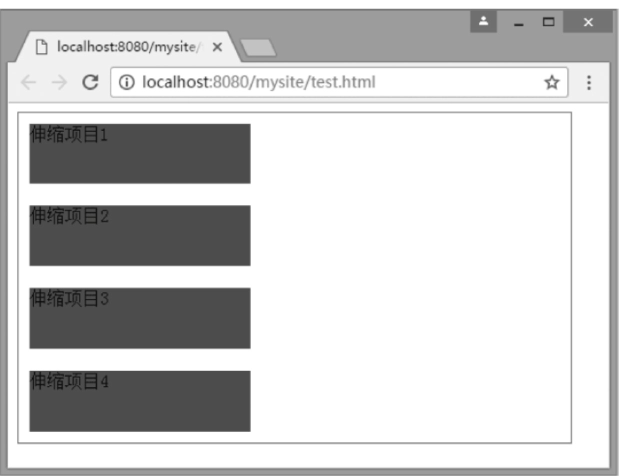
☑　column-reverse：对齐方式与 column 相反。

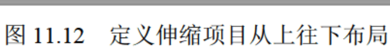
【示例】在上节示例的基础上，本例将设计一个伸缩容器，其中包含 4 个伸缩项目，然后定义伸缩项目从上往下排列。代码如下，演示效果如图 11.12 所示。

```
<style type="text/css">
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-flex-direction: column;
    flex-direction: column;
    width: 500px;height: 300px;border: solid 1px red;
}
.flex-item {
    background-color: blue;
    width: 200px; height: 200px;
    margin: 10px;
}
</style>
```








图 11.12　定义伸缩项目从上往下布局

11.3.4　设置行数



flex-wrap 定义伸缩容器是单行还是多行显示伸缩项目，侧轴的方向决定了新行堆放的方向。具体语法格式如下。

```
flex-wrap：nowrap | wrap | wrap-reverse
```

取值说明如下。

☑　nowrap：flex 容器为单行。该情况下 flex 子项可能会溢出容器。

☑　wrap：flex 容器为多行。该情况下 flex 子项溢出的部分会被放置到新行，子项内部会发生断行。

☑　wrap-reverse：反转 wrap 排列。

【示例】在上面示例的基础上，本示例将设计一个伸缩容器，其中包含 4 个伸缩项目，然后定义伸缩项目多行排列。代码如下，演示效果如图 11.13 所示。

```
<style type="text/css">
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-flex-wrap: wrap;
    flex-wrap: wrap;
    width: 500px; height: 300px;border: solid 1px red;
}
.flex-item {
    background-color: blue;
    width: 200px; height: 200px;
```




Note

```
        margin: 10px;
    }
</style>
```

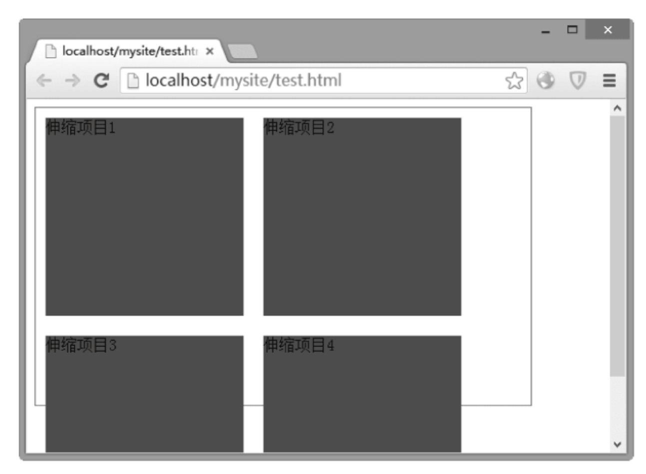


图 11.13　定义伸缩项目多行布局

【补充】

flex-flow 属性是 flex-direction 和 flex-wrap 属性的复合属性，适用于伸缩容器。该属性可以同时定义伸缩容器的主轴和侧轴。其默认值为 row nowrap。具体语法如下。

```
flex-flow：<' flex-direction '> || <' flex-wrap '>
```

取值说明如下。

☑ <' flex-direction'>：定义弹性盒子元素的排列方向。

☑ <' flex-wrap'>：控制 flex 容器是单行还是多行。

视 频 讲 解

11.3.5　设置对齐方式

1. 主轴对齐

justify-content 定义伸缩项目沿着主轴线对齐，该属性适用于伸缩容器。具体语法如下。



```
justify-content：flex-start | flex-end | center | space-between | space-around
```

取值说明如下。

☑ flex-star：为默认值，伸缩项目向一行的起始位置靠齐。

☑ flex-end：伸缩项目向一行的结束位置靠齐。

☑ center：伸缩项目向一行的中间位置靠齐。

☑ space-between：伸缩项目会平均地分布在行里。第一个伸缩项目在一行中的开始位置，最后一个伸缩项目在一行中的终点位置。

☑ space-around：伸缩项目会平均地分布在行里，两端保留一半的空间。



上述取值比较效果如图 11.14 所示。

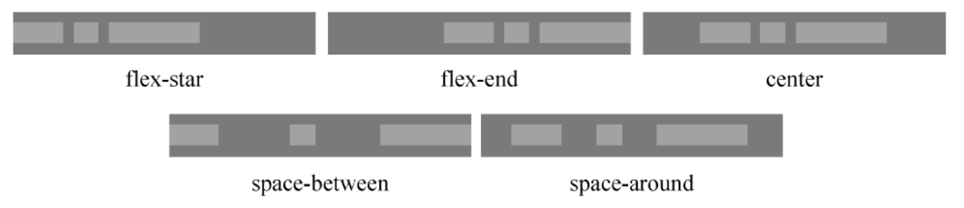


图 11.14　主轴对齐示意图

2. 侧轴对齐

align-items 定义伸缩项目在侧轴上的对齐方式，该属性适用于伸缩容器。具体语法如下。

```
align-items：flex-start | flex-end | center | baseline | stretch
```

取值说明如下。

☑ flex-start：伸缩项目在侧轴起点边的外边距紧靠住该行在侧轴起始的边。
☑ flex-end：伸缩项目在侧轴终点边的外边距靠住该行在侧轴终点的边。
☑ center：伸缩项目的外边距盒在该行的侧轴上居中放置。
☑ baseline：伸缩项目根据它们的基线对齐。
☑ stretch：默认值，伸缩项目拉伸填充整个伸缩容器。此值会使项目的外边距盒的尺寸在遵照 min/max-width/height 属性的限制下尽可能接近所在行的尺寸。

上述取值比较效果如图 11.15 所示。

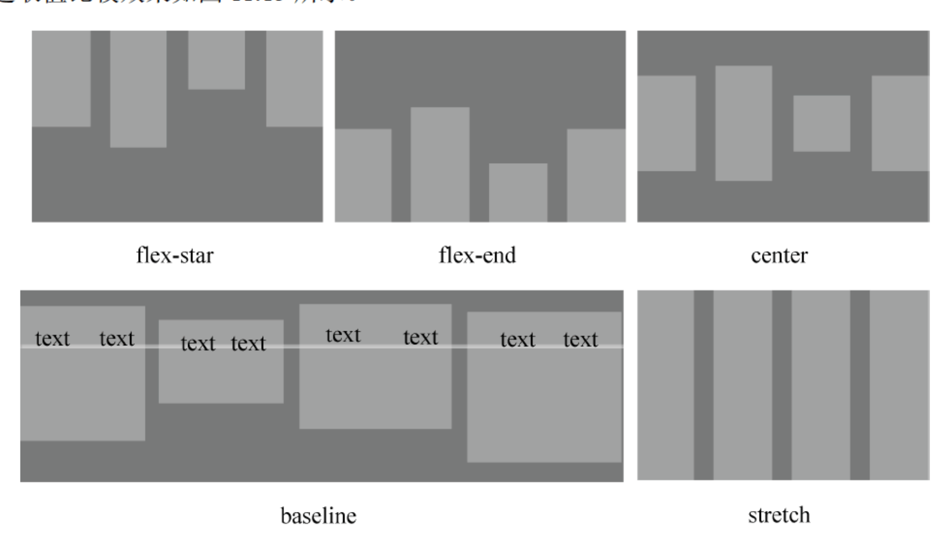


图 11.15　侧轴对齐示意图

3. 伸缩行对齐

align-content 定义伸缩行在伸缩容器里的对齐方式，该属性适用于伸缩容器。类似于伸缩项目在主轴上使用 justify-content 属性，但本属性在只有一行的伸缩容器上没有效果。具体语法如下。



```
align-content：flex-start | flex-end | center | space-between | space-around | stretch
```



Note

取值说明如下。

☑ flex-start：各行向伸缩容器的起点位置堆叠。

☑ flex-end：各行向伸缩容器的结束位置堆叠。

☑ center：各行向伸缩容器的中间位置堆叠。

☑ space-between：各行在伸缩容器中平均分布。

☑ space-around：各行在伸缩容器中平均分布，在两边各有一半的空间。

☑ stretch：默认值，各行将会伸展以占用剩余的空间。

上述取值比较效果如图 11.16 所示。

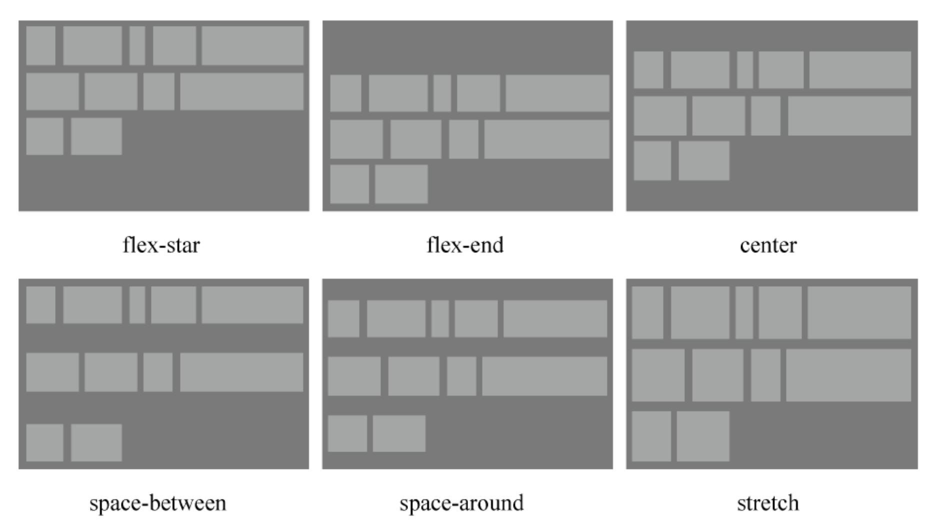


图 11.16　伸缩行对齐示意图

【示例】本示例以上面的示例为基础，定义伸缩行在伸缩容器中居中显示。代码如下，演示效果如图 11.17 所示。

```
<style type="text/css">
.flex-container {
    display: -webkit-flex;
    display: flex;
    -webkit-flex-wrap: wrap;
    flex-wrap: wrap;
    -webkit-align-content: center;
    align-content: center;
    width: 500px; height: 300px;border: solid 1px red;
}
.flex-item {
```





```
        background-color: blue;
        width: 200px; height: 200px;
        margin: 10px;
    }
</style>
```

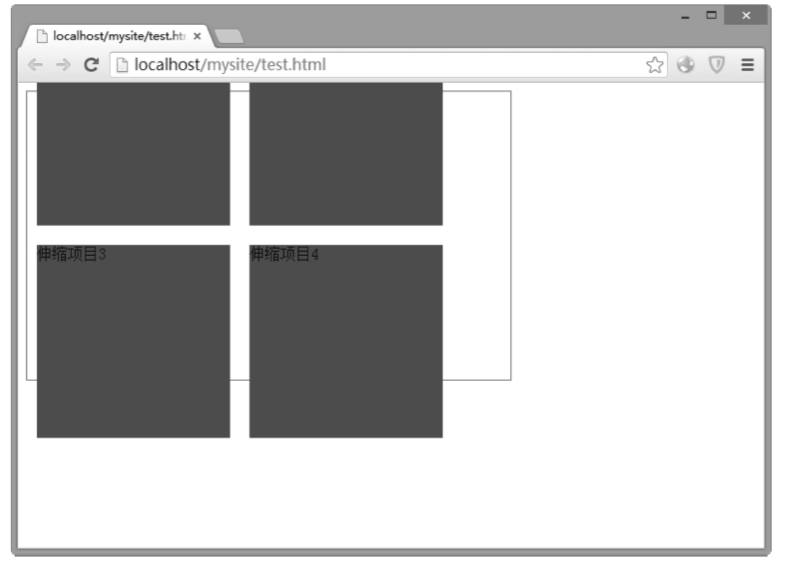


图 11.17　定义伸缩行居中对齐

11.3.6　设置伸缩项目



伸缩项目都有一个主轴长度（Main Size）和一个侧轴长度（Cross Size）。主轴长度是伸缩项目在主轴上的尺寸，侧轴长度是伸缩项目在侧轴上的尺寸。一个伸缩项目的宽或高取决于伸缩容器的轴，可能就是它的主轴长度或侧轴长度。下面的属性适用于伸缩项目，可以调整伸缩项目的行为。

1. 显示位置

order 属性可以控制伸缩项目在伸缩容器中的显示顺序，具体语法如下。

```
order: <integer>
```

<integer>用整数值来定义排列顺序，数值小的排在前面。可以为负值。

2. 扩展空间

flex-grow 可以定义伸缩项目的扩展能力，决定伸缩容器剩余空间按比例应扩展多少。具体语法如下。

```
flex-grow: <number>
```

<number>用数值来定义扩展比率。不允许有负值，默认值为 0。



Note

如果所有伸缩项目的 flex-grow 均设置为 1，那么每个伸缩项目将设置为一个大小相等的剩余空间。如果设置其中一个伸缩项目的 flex-grow 为 2，那么这个伸缩项目所占的剩余空间是其他伸缩项目所占剩余空间的两倍。

3. 收缩空间

flex-shrink 可以定义伸缩项目收缩的能力，与 flex-grow 功能相反。具体语法如下。

```
flex-shrink: <number>
```

<number>用数值来定义收缩比率。不允许有负值，默认值为 1。

4. 伸缩比率

flex-basis 可以设置伸缩基准值，剩余的空间按比率进行伸缩。具体语法如下。

```
flex-basis：<length> | <percentage> | auto | content
```

取值说明如下。

☑ <length>：用长度值来定义宽度。不允许有负值。

☑ <percentage>：用百分比来定义宽度。不允许有负值。

☑ auto：无特定宽度值，取决于其他属性值。

☑ content：基于内容自动计算宽度。

【补充】

flex 是 flex-grow、flex-shrink 和 flex-basis 3 个属性的复合属性，该属性适用于伸缩项目。其中第 2 个和第 3 个参数（flex-shrink、flex-basis）是可选参数。默认值为“0 1 auto”。具体语法如下。

```
flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]
```

5. 对齐方式

align-self 用来在单独的伸缩项目上覆写默认的对齐方式。具体语法如下。

```
align-self：auto | flex-start | flex-end | center | baseline | stretch
```

属性值与 align-items 的属性值相同。

【示例 1】以上面的示例为基础，定义伸缩项目在当前位置向右错移一个位置，其中第 1 个项目位于第 2 项目的位置，第 2 个项目位于第 3 个项目的位置，最后一个项目移到第 1 个项目的位置。代码如下，演示效果如图 11.18 所示。

```
<style type="text/css">
.flex-container {
    display: -webkit-flex;
    display: flex;
    width: 500px; height: 300px;border: solid 1px red;
}
```





```
.flex-item { background-color: blue; width: 200px; height: 200px; margin: 10px;}
.flex-item:nth-child(0){
    -webkit-order: 4;
    order: 4;
}
.flex-item:nth-child(1){
    -webkit-order: 1;
    order: 1;
}
.flex-item:nth-child(2){
    -webkit-order: 2;
    order: 2;
}
.flex-item:nth-child(3){
    -webkit-order: 3;
    order: 3;
}
</style>
```

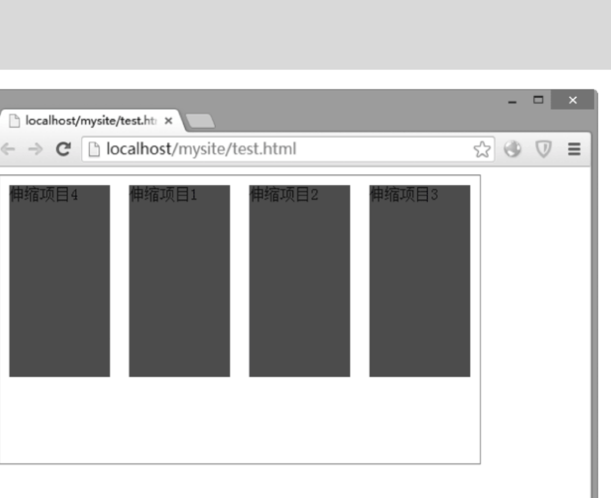


图 11.18　定义伸缩项目错位显示

【示例 2】margin: auto;在伸缩盒中具有强大的功能，一个“auto”的 margin 会合并剩余的空间。它可以用来把伸缩项目挤到其他位置。本示例利用 margin-right: auto;定义包含的项目居中显示，代码如下，效果如图 11.19 所示。

```
<style type="text/css">
.flex-container {
    display: -webkit-flex;
```




```
        display: flex;
        width: 500px; height: 300px; border: solid 1px red;
}
.flex-item {
        background-color: blue; width: 200px; height: 200px;
        margin: auto;
}
</style>

<div class="flex-container">
        <div class="flex-item">伸缩项目</div>
</div>
```

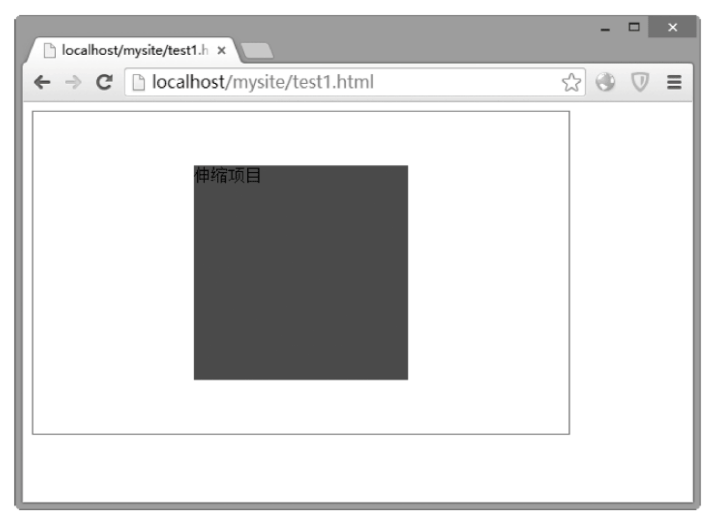


图 11.19　定义伸缩项目居中显示

11.4　伸缩盒版本比较和兼容

本节将重点介绍伸缩盒模型的老版本、混合版本和新版本之间的用法差异及兼容方法，以便用户设计在不同浏览器上都能正确显示的弹性布局。

11.4.1　版本比较和兼容方法

CSS3 伸缩盒布局还在不断发展中，并不断升级，大致经历了 3 个阶段，并逐步达到稳定，主流浏览器对新版本也开始完整地支持。

☑　2009 年版本（旧版本）：display:box;。

☑　2011 年版本（混合版本）：display:flexbox;。



☑　2012 年版本（新版本）：display:flex;。

如果把 Flexbox 新语法、旧语法和混合语法混合在一起使用，就可以让浏览器得到完美地展示。当然，在使用 Flexbox 时，应该考虑不同浏览器的私有属性，如 Chrome 要添加前缀-webkit-，Firefox 要添加前缀-moz-等。



1．浏览器支持状况

各主流浏览器对 Flexbox 规范版本的支持如表 11.1 所示。

表 11.1　浏览器对规范版本的支持

规 范 版 本	IE	Opera	Firefox	Chrome	Safari
新版本（标准版）	11	12.10+ *	22	29+、21–28 (-webkit-)	
混合版	10 (-ms-)				
老版本			3–21 (-moz-)	<21 (-webkit-)	3–6 (-webkit-)

2．开启 Flexbox

不同 Flexbox 版本定义一个元素为伸缩容器的方法比较如表 11.2 所示。

表 11.2　比较启动 Flexbox

规 范 版 本	属 性 名 称	块伸缩容器	内联伸缩容器
新版本（标准版）	display	flex	inline-flex
混合版	display	flexbox	inline-flexbox
老版本	display	box	inline-box

3．主轴对齐方式

不同 Flexbox 版本指定伸缩项目沿主轴对齐方式的取值比较如表 11.3 所示。

表 11.3　比较主轴对齐方式

规 范 版 本	属 性 名 称	start	center	end	justify	distribute
新版本（标准版）	justify-content	flex-start	center	flex-end	space-between	space-around
混合版	flex-pack	start	center	end	justify	distribute
老版本	box-pack	start	center	end	justify	N/A

注：
- start：开始位置。
- center：中间位置。
- end：结束位置。
- justify：两端对齐。
- distribute：均匀对齐。
- N/A：表示不适用的意思。

4．侧轴对齐方式

不同 Flexbox 版本指定伸缩项目沿侧轴对齐方式的取值比较如表 11.4 所示。






表 11.4　比较侧轴对齐方式

规 范 版 本	属 性 名 称	start	center	end	baseline	stretch
新版本（标准版）	align-items	flex-start	center	flex-end	baseline	stretch
混合版	flex-align	start	center	end	baseline	stretch
老版本	box-align	start	center	end	baseline	stretch

注：
- baseline：基线对齐。
- stretch：伸展对齐。

5. 单个伸缩项目侧轴对齐方式

不同 Flexbox 版本指定单个伸缩项目沿侧轴对齐方式的取值比较如表 11.5 所示。

表 11.5　比较单个伸缩项目侧轴对齐方式

规 范 版 本	属 性 名 称	auto	start	center	end	baseline	stretch
新版本（标准版）	align-self	auto	flex-start	center	flex-end	baseline	stretch
混合版	flex-item-align	auto	start	center	end	baseline	stretch
老版本	N/A						

6. 伸缩项目行对齐方式

不同 Flexbox 版本指定伸缩项目行在侧轴的对齐方式的取值比较如表 11.6 所示。

表 11.6　比较伸缩项目侧轴对齐方式

规 范 版 本	属 性 名 称	start	center	end	justify	distribute	stretch
新版本（标准版）	align-content	flex-start	center	flex-end	space-between	space-around	stretch
混合版	flex-line-pack	start	center	end	justify	distribute	stretch
老版本	N/A						

注意： 只有伸缩项目有多行时才生效，这种情况只针对伸缩容器设置了 flex-wrap 为 wrap，并且没有足够的空间把伸缩项目放在同一行中。这将对每一行起作用而不是对每一个伸缩项目起作用。

7. 显示顺序

不同 Flexbox 版本指定伸缩项目的显示顺序的取值比较如表 11.7 所示。

表 11.7　比较显示顺序

规 范 版 本	属 性 名 称	属 性 值
新版本（标准版）	order	<number>
混合版	flex-order	<number>
老版本	box-ordinal-group	<integer>





8. 伸缩性

不同 Flexbox 版本指定伸缩项目如何伸缩比较如表 11.8 所示。

表 11.8　比较伸缩性

规 范 版 本	属 性 名 称	属　性　值
新版本（标准版）	flex	none | [<flex-grow> <flex-shrink>? || <flex-basis>]
混合版	flex	none | [[<pos-flex> <neg-flex>?] || <preferred-size>]
老版本	box-flex	<number>

flex 属性在微软的草案与新标准或多或少不一样。它们都转换成标准缩写版本，属性值为 flex-grow、flex-shrink 和 flex-basis。主要区别在于：flex-shrink（以前称为负 flex）的默认值为 1。这意味着伸缩项目默认不能收缩。以前，空间不足使用 flex-shrink 比例来收缩伸缩项目，但现在可以在 flex-basis 的基础上配合 flex-shrink 来收缩伸缩项目。

9. 伸缩流

不同 Flexbox 版本指定伸缩容器主轴的伸缩流方向比较如表 11.9 所示。

表 11.9　比较伸缩流

规 范 版 本	属 性 名 称	Horizontal	Reversed horizontal	Vertical	Reversed vertical
新版本（标准版）	flex-direction	row	row-reverse	column	column-reverse
混合版	flex-direction	row	row-reverse	column	column-reverse
老版本	box-orient	horizontal	horizontal	vertical	vertical
	box-direction	normal	reverse	normal	reverse

在旧版本规范中，将 box-direction 属性设置为 reverse 和在新版本中设置 row-reverse 或 column-reverse 得到的效果相同。如果想要的效果是 row 或 column，可以省略不设置，因为 normal 是默认的初始值。

当设置 direction 为 reverse 时，主轴就翻转。例如，当使用“ltr”书写模式指定 row-reverse 时，所有伸缩项目会从右向左排列。类似的，column-reverse 将会使所有伸缩项目从下向上排列，来代替从上往下排列。

在老版本中，需要使用 box-orient 来设置书写模式的方向。当使用“ltr”模式时，horizontal 可用 inline-axis，vertical 可用 block-axis。如果使用的是一个自上而下的书写模式，如东亚传统的书写模式，这些值就会翻转。

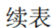
10. 换行

不同 Flexbox 版本指定伸缩项目是否沿着侧轴排列比较如表 11.10 所示。

表 11.10　比较换行

规 范 版 本	属 性 名 称	No wrapping	Wrapping	Reversed wrap
新版本（标准版）	flex-wrap	nowrap	wrap	wrap-reverse




续表


Note

规 范 版 本	属 性 名 称	No wrapping	Wrapping	Reversed wrap
混合版	flex-wrap	nowrap	wrap	wrap-reverse
老版本	box-lines	single	multiple	N/A

wrap-reverse 让伸缩项目在侧轴上进行 start 和 end 翻转，所以，如果伸缩项目在水平排列，伸缩项目不会翻转到一个新的线下面，它会翻转到一个新的线上面。简单理解就是伸缩项目只是上下或前后翻转。


视 频 讲 解

11.4.2 案例：设计 3 栏页面

本案例根据上节介绍的方法，使用不同版本语法，设计一个兼容不同设备和浏览器的弹性页面，演示效果如图 11.20 所示。

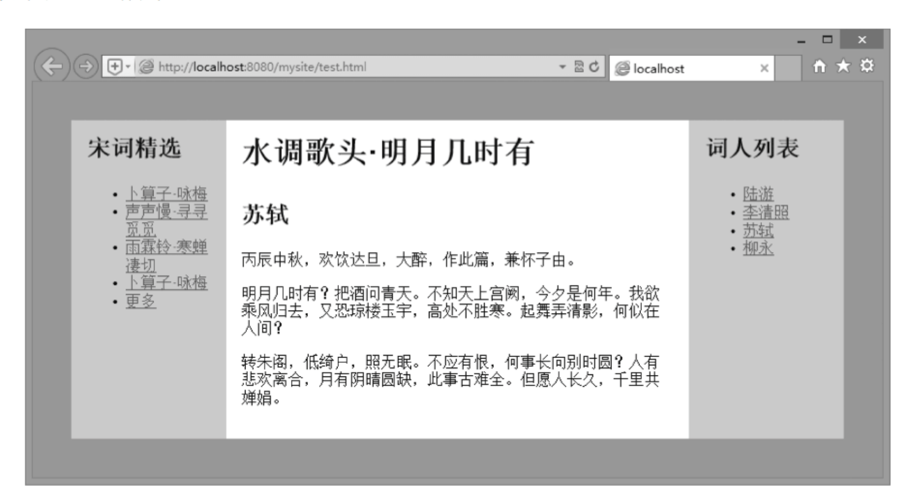


图 11.20　3 栏页面效果

案例主要代码如下。

```
<style type="text/css">
.page-wrap {
    display: -webkit-box;              /* 2009 版  - iOS 6-, Safari 3.1-6 */
    display: -moz-box;                 /* 2009 版  - Firefox 19-（存在缺陷）  */
    display: -ms-flexbox;              /* 2011 版  - IE 10 */
    display: -webkit-flex;             /*  最新版  - Chrome */
    display: flex;                     /*  最新版  - Opera 12.1, Firefox 20+ */
}
.main-content {
    -webkit-box-ordinal-group: 2;      /* 2009 版  - iOS 6-, Safari 3.1-6 */
    -moz-box-ordinal-group: 2;         /* 2009 版  - Firefox 19- */
```





```
    -ms-flex-order: 2;                  /* 2011 版 - IE 10 */
    -webkit-order: 2;                   /* 最新版 - Chrome */
    order: 2;                           /* 最新版 - Opera 12.1, Firefox 20+ */
    width: 60%;                         /* 不会自动伸缩，其他列将占据空间 */
    -moz-box-flex: 1;                   /* 如果没有该声明，主内容（60%）会伸展到最宽的段落，就像是
                                        段落设置了 white-space:nowrap */
    background: white;
}
.main-nav {
    -webkit-box-ordinal-group: 1;       /* 2009 版 - iOS 6-, Safari 3.1-6 */
    -moz-box-ordinal-group: 1;          /* 2009 版 - Firefox 19- */
    -ms-flex-order: 1;                  /* 2011 版 - IE 10 */
    -webkit-order: 1;                   /* 最新版 - Chrome */
    order: 1;                           /* 最新版 - Opera 12.1, Firefox 20+ */
    -webkit-box-flex: 1;                /* 2009 版 - iOS 6-, Safari 3.1-6 */
    -moz-box-flex: 1;                   /* 2009 版 - Firefox 19- */
    width: 20%;                         /* 2009 版语法，否则将崩溃 */
    -webkit-flex: 1;                    /* Chrome */
    -ms-flex: 1;                        /* IE 10 */
    flex: 1;                            /* 最新版 - Opera 12.1, Firefox 20+ */
    background: #ccc;
}
.main-sidebar {
    -webkit-box-ordinal-group: 3;       /* 2009 版 - iOS 6-, Safari 3.1-6 */
    -moz-box-ordinal-group: 3;          /* 2009 版 - Firefox 19- */
    -ms-flex-order: 3;                  /* 2011 版 - IE 10 */
    -webkit-order: 3;                   /* 最新版 - Chrome */
    order: 3;                           /* 最新版- Opera 12.1, Firefox 20+ */
    -webkit-box-flex: 1;                /* 2009 版 - iOS 6-, Safari 3.1-6 */
    -moz-box-flex: 1;                   /* Firefox 19- */
    width: 20%;                         /* 2009 版，否则将崩溃 */
    -ms-flex: 1;                        /* 2011 版 - IE 10 */
    -webkit-flex: 1;                    /* 最新版 - Chrome */
    flex: 1;                            /* 最新版 - Opera 12.1, Firefox 20+ */
    background: #ccc;
}
.main-content, .main-sidebar, .main-nav { padding: 1em; }
body {padding: 2em; background: #79a693;}
```




```
* {
    -webkit-box-sizing: border-box;
    -moz-box-sizing: border-box;
    box-sizing: border-box;}
h1, h2 {
    font: bold 2em Sans-Serif;
    margin: 0 0 1em 0;}
h2 { font-size: 1.5em; }
p { margin: 0 0 1em 0; }
</style>

<div class="page-wrap">
    <section class="main-content">
        <h1>水调歌头·明月几时有</h1>
        ……
    </section>
    <nav class="main-nav">
        <h2>宋词精选</h2>
        ……
    </nav>
    <aside class="main-sidebar">
        <h2>词人列表</h2>
        ……
    </aside>
</div>
```

页面被包裹在类名为 page-wrap 的容器中，容器包含 3 个子模块。现在将容器定义为伸缩容器，此时每个子模块自动变成了伸缩项目，代码如下。

```
<div class="page-wrap">
    <section class="main-content"> </section>
    <nav class="main-nav"></nav>
    <aside class="main-sidebar"></aside>
</div>
```

本例设计各列在一个伸缩容器中显示上下文，只有这样这些元素才能直接成为伸缩项目，它们之前是什么没有关系，只要现在是伸缩项目即可。

本例把 Flexbox 旧的语法、中间混合语法和最新的语法混在一起使用，它们的顺序很重要。display 属性本身并不添加任何浏览器前缀，用户需要确保老语法不要覆盖新语法，让浏览器同时支持，代





码如下。

```
.page-wrap {
    display: -webkit-box;          /* 2009 版 - iOS 6-, Safari 3.1-6 */
    display: -moz-box;             /* 2009 版 - Firefox 19-（存在缺陷） */
    display: -ms-flexbox;          /* 2011 版 - IE 10 */
    display: -webkit-flex;         /* 最新版 - Chrome */
    display: flex;                 /* 最新版 - Opera 12.1，Firefox 20+ */
}
```

整个页面包含 3 列，设计一个 20%、60%、20%的网格布局。首先设置主内容区域宽度为 60%，然后设置侧边栏来填补剩余的空间。同样把新旧语法混在一起使用，代码如下。

```
.main-content {
    -webkit-box-ordinal-group: 2;  /* 2009 版 - iOS 6-, Safari 3.1-6 */
    -moz-box-ordinal-group: 2;     /* 2009 版 - Firefox 19- */
    -ms-flex-order: 2;             /* 2011 版 - IE 10 */
    -webkit-order: 2;              /* 最新版 - Chrome */
    order: 2;                      /* 最新版 - Opera 12.1，Firefox 20+ */
    width: 60%;                    /* 不会自动伸缩，其他列将占据空间 */
    -moz-box-flex: 1;              /* 如果没声明，Firefox 19-将溢出 h，覆盖宽度 */
    background: white;
}
```

在新语法中，没有必要给边栏设置宽度，因为它们同样会使用 20%的比例填充剩余的 40%空间。但是，如果不显示设置宽度，在老的语法下会直接崩溃。

完成初步布局之后，需要重新排列顺序。这里设计主内容排列在中间，但在源码之中，它是排列在第一的位置。使用 Flexbox 可以非常容易实现，但是用户需要把 Flexbox 几种不同的语法混在一起使用，代码如下。

```
.main-content {
    -webkit-box-ordinal-group: 2;
    -moz-box-ordinal-group: 2;
    -ms-flex-order: 2;
    -webkit-order: 2;
    order: 2;
}
.main-nav {
    -webkit-box-ordinal-group: 1;
    -moz-box-ordinal-group: 1;
```





Note

```
        -ms-flex-order: 1;
        -webkit-order: 1;
        order: 1;
}
.main-sidebar {
        -webkit-box-ordinal-group: 3;
        -moz-box-ordinal-group: 3;
        -ms-flex-order: 3;
        -webkit-order: 3;
        order: 3;
}
```



视 频 讲 解

11.4.3 案例：设计 3 行 3 列应用

本例借助 Flexbox 伸缩盒布局，设计页面呈现 3 行 3 列布局样式，同时能够根据窗口自适应调整各自空间，以满屏显示，效果如图 11.21 所示。

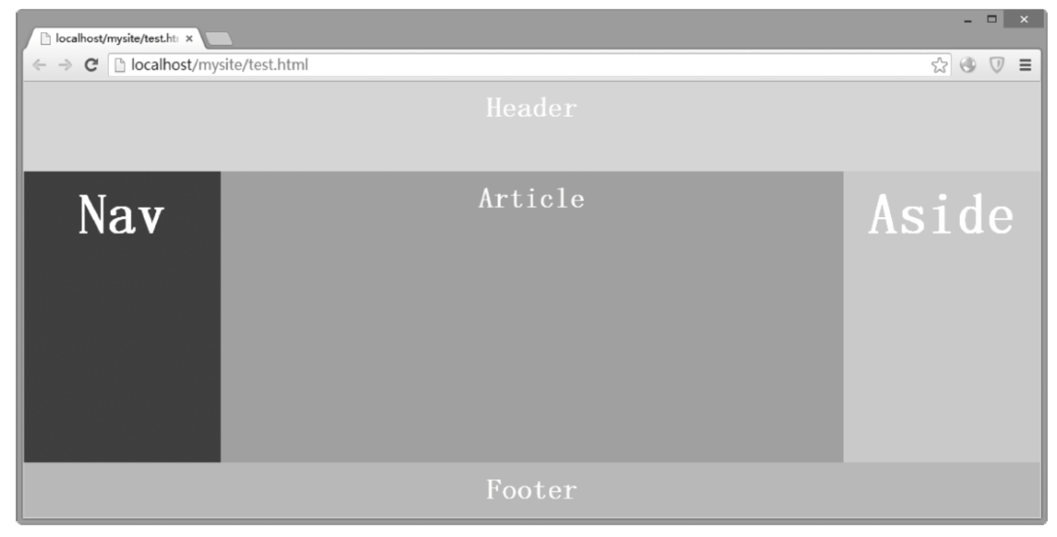


图 11.21　3 行 3 列布局样式效果

页面主要代码如下。

```
<style type="text/css">
/*基本样式*/
* {margin: 0; padding: 0;
        -moz-box-sizing: border-box;
        -webkit-box-sizing: border-box;
        box-sizing: border-box;
```





```
}
html, body {height: 100%; color: #fff;}
body { min-width: 100%; }
header, section, nav, aside, footer { display: block; text-align:center; font-size:2em; font-weight:bold; }
header {/*页眉框样式：限高、限宽*/
    background-color: hsla(200,10%,70%,.5);
    min-height: 100px; padding: 10px 20px;
    min-width: 100%;
}
/*主体区域框样式：满宽显示*/
section {min-width: 100%;}
/*导航框样式：固定宽度*/
nav {background-color: hsla(300,60%,20%,.9);padding: 1%;width: 220px;}
/*文档栏样式*/
article {background-color: hsla(120,50%,50%,.9); padding: 1%;}
/*侧边栏样式：弹性宽度*/
aside {background-color: hsla(20,80%,80%,.9); padding: 1%;width: 220px;}
footer {/*页脚样式：限高、限宽*/
    background-color: hsla(250,50%,80%,.9);
    min-height: 60px; padding: 1%;
    min-width: 100%;
}
body {/*flexbox 样式*/
    /*设置 body 为伸缩容器*/
    display: -webkit-box;/*老版本：iOS 6-, Safari 3.1-6*/
    display: -moz-box;/*老版本：Firefox 19- */
    display: -ms-flexbox;/*混合版本：IE10*/
    display: -webkit-flex;/*新版本：Chrome*/
    display: flex;/*标准规范：Opera 12.1, Firefox 20+*/
    /*伸缩项目换行*/
    -moz-box-orient: vertical;
    -webkit-box-orient: vertical;
    -moz-box-direction: normal;
    -moz-box-direction: normal;
    -moz-box-lines: multiple;
    -webkit-box-lines: multiple;
    -webkit-flex-flow: column wrap;
    -ms-flex-flow: column wrap;
```





Note

```
        flex-flow: column wrap;
    }
    section {/*实现 stick footer 效果*/
        display: -moz-box;
        display: -webkit-box;
        display: -ms-flexbox;
        display: -webkit-flex;
        display: flex;
        -webkit-box-flex: 1;
        -moz-box-flex: 1;
        -ms-flex: 1;
        -webkit-flex: 1;
        flex: 1;
        -moz-box-orient: horizontal;
        -webkit-box-orient: horizontal;
        -moz-box-direction: normal;
        -webkit-box-direction: normal;
        -moz-box-lines: multiple;
        -webkit-box-lines: multiple;
        -ms-flex-flow: row wrap;
        -webkit-flex-flow: row wrap;
        flex-flow: row wrap;
        -moz-box-align: stretch;
        -webkit-box-align: stretch;
        -ms-flex-align: stretch;
        -webkit-align-items: stretch;
        align-items: stretch;
    }
    article {/*文章区域伸缩样式*/
        -moz-box-flex: 1;
        -webkit-box-flex: 1;
        -ms-flex: 1;
        -webkit-flex: 1;
        flex: 1;
        -moz-box-ordinal-group: 2;
        -webkit-box-ordinal-group: 2;
        -ms-flex-order: 2;
        -webkit-order: 2;
```



```
        order: 2;
}
aside {/*侧边栏伸缩样式*/
        -moz-box-ordinal-group: 3;
        -webkit-box-ordinal-group: 3;
        -ms-flex-order: 3;
        -webkit-order: 3;
        order: 3;
}
</style>

<header>Header</header>
<section>
        <article>Article</article>
        <nav>Nav</nav>
        <aside>Aside</aside>
</section>
<footer>Footer</footer>
```

11.5　在 线 练 习

Flexbox 3 个不同版本的规范对应着不同的实现。需要关注哪个版本，取决于需要支持的浏览器。



在 线 练 习



使用 CSS3 设计动态样式

CSS3 动画包括过渡动画和关键帧动画，主要通过改变 CSS 属性值来模拟实现。本章将详细介绍 Transform、Transitions 和 Animations 三大功能模块，其中 Transform 实现对网页对象的变形操作，Transitions 实现 CSS 属性的过渡变化，Animations 实现 CSS 样式分步式演示效果。

【学习要点】

- 设计对象变形操作。
- 设计过渡样式。
- 设计关键帧动画。
- 能够灵活使用 CSS3 动画设计页面特效。





12.1　CSS3 变形

2012 年 9 月，W3C 发布了 CSS3 变形工作草案。CSS3 变形允许 CSS 把元素转变为 2D 或 3D 空间，这个草案包括 CSS3 2D 变形和 CSS3 3D 变形。

权威参考：http://www.w3.org/TR/css-transforms-1/。

12.1.1　认识 Transform

CSS3 变形是多种效果的集合，如旋转、缩放、平移和倾斜等，每个效果都被称为变形函数，它们可以操控元素发生旋转、缩放、平移和倾斜等变化。在 CSS3 之前，实现类似的效果都需要图片、Flash 或 JavaScript 才能完成。而使用纯 CSS 来完成这些变形则无须加载这些额外的文件，提高了开发效率和页面的执行效率。

CSS3 变形包括 3D 变形和 2D 变形，3D 变形使用基于 2D 变形的相同属性，如果了解了 2D 变形，会发现 3D 变形与 2D 变形的功能类似。

CSS 2D Transform 获得了各主流浏览器的支持，但是 CSS 3D Transform 支持程度不是很完善。考虑到浏览器兼容性，3D 变形在实际应用时应添加私有属性，并且个别属性在某些主流浏览器中并未得到很好的支持，简单说明如下。

☑ 在 IE10+中，3D 变形部分属性未得到很好的支持。

☑ Firefox10.0～Firefox15.0 版本的浏览器，在使用 3D 变形时需要添加私有属性-moz-，但从 Firefox16.0+版本开始无须添加浏览器私有属性。

☑ Chrome 12.0+版本中使用 3D 变形时需要添加私有属性-webkit-。

☑ Safari 4.0+版本中使用 3D 变形时需要添加私有属性-webkit-。

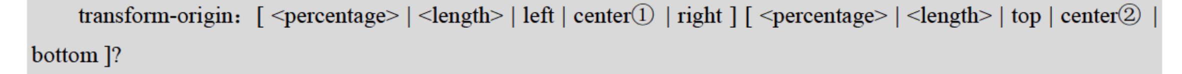
☑ Opera15.0+版本才开始支持 3D 变形，使用时需要添加私有属性-webkit-。

☑ 移动设备中，iOS Safari 3.2+、Android Browser 3.0+、Blackberry Browser 7.0+、Opera Mobile 24.0+、Chrome for Android 25.0+都支持 3D 变形，但在使用时需要添加私有属性-webkit-；Firefox for Android 19.0+支持 3D 变形，且无须添加浏览器私有属性。

12.1.2　设置原点



CSS 变形的原点默认为对象的中心点（50% 50%），使用 transform-origin 属性可以重新设置新的变形原点，语法格式如下。



```
transform-origin：[ <percentage> | <length> | left | center① | right ] [ <percentage> | <length> | top | center② | bottom ]?
```

取值简单说明如下。

☑ <percentage>：用百分比指定坐标值。可以为负值。

☑ <length>：用长度值指定坐标值。可以为负值。

☑ left：指定原点的横坐标为 left。





Note

☑ center①：指定原点的横坐标为 center。

☑ right：指定原点的横坐标为 right。

☑ top：指定原点的纵坐标为 top。

☑ center②：指定原点的纵坐标为 center。

☑ bottom：指定原点的纵坐标为 bottom。

【示例】通过重置变形原点，可以设计不同的变形效果。在本示例中以图像的右上角为原点逆时针旋转图像 45°，则比较效果如图 12.1 所示。

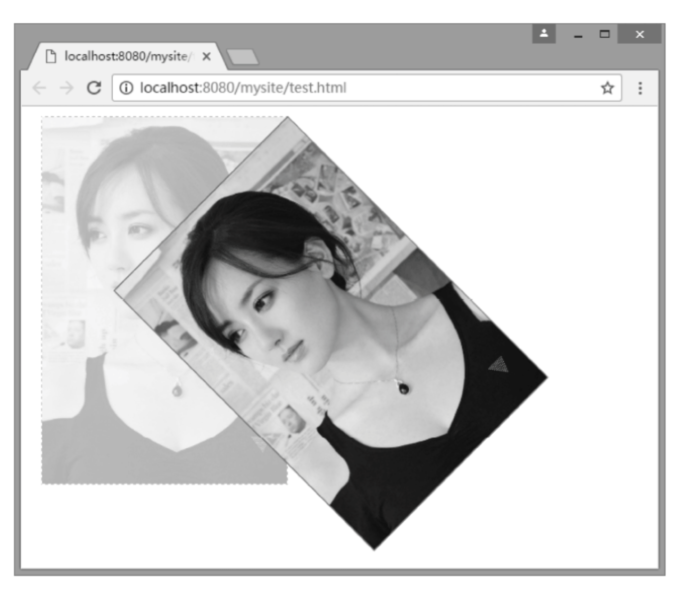


图 12.1　自定义旋转原点效果

示例代码如下。

```
<style type="text/css">
img {/* 固定两幅图像相同大小和相同显示位置 */
    position: absolute;
    left: 20px;
    top: 10px;
    width: 170px;
    width: 250px;
}
img.bg {/* 设置第 1 幅图像作为参考 */
    opacity: 0.3;
    border: dashed 1px red;
}
img.change {/* 变形第 2 幅图像 */
    border: solid 1px red;
```



```
        transform-origin: top right;                /*以右上角为原点进行变形*/
        transform: rotate(-45deg);                  /*逆时针旋转 45°  */
    }
    </style>

    <img class="bg" src="images/1.jpg">
    <img class="change" src="images/1.jpg">
```







12.1.3　2D 旋转

rotate()函数能够在 2D 空间内旋转对象，语法格式如下。

```
rotate(<angle>)
```

参数 angle 表示角度值，取值单位可以是：度，如 90deg（90°，一圈 360°）；梯度，如 100grad（相当于 90°，360°等于 400grad）；弧度，如 1.57rad（约等于 90°，360°等于 2π）；圈，如 0.25turn（等于 90°，360°等于 1turn）。

【示例】 以 12.1.2 节示例为基础，本示例按默认原点逆时针旋转图像 45°，代码如下，演示效果如图 12.2 所示。

```
img.change {
    border: solid 1px red;
    transform: rotate(-45deg);
}
```

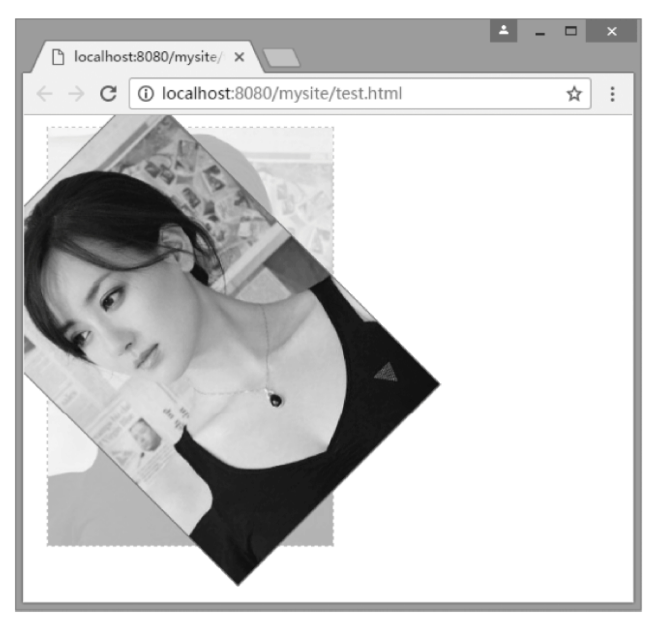


图 12.2　旋转效果





Note



视频讲解

12.1.4 2D 缩放

scale()函数能够缩放对象大小，语法格式如下。

```
scale(<number>[, <number>])
```

该函数包含两个参数值，分别用来定义宽和高的缩放比例，取值简单说明如下。

☑ 如果取值为正数，则基于指定的宽度和高度放大或缩小对象。

☑ 如果取值为负数，则不会缩小元素，而是翻转元素（如文字被翻转），然后再缩放元素。

☑ 如果取值为小于 1 的小数（如 0.5），可以缩小元素。

☑ 如果第 2 个参数省略，则第 2 个参数等于第 1 个参数值。

【示例】继续以 12.1.2 节示例为基础，本示例按默认原点把图像缩小为原来的二分之一，代码如下，演示效果如图 12.3 所示。

```
img.change {
    border: solid 1px red;
    transform: scale(0.5);
}
```

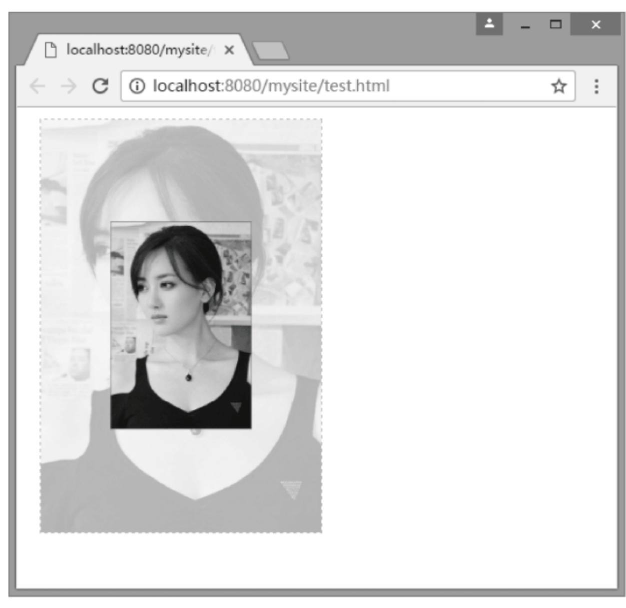


图 12.3　缩小对象的效果



视频讲解

12.1.5 2D 平移

translate()函数能够平移对象的位置，语法格式如下。

```
translate(<translation-value>[, <translation-value>])
```





该函数包含两个参数值，分别用来定义对象在 x 轴和 y 轴相对于原点的偏移距离。如果省略参数，则默认值为 0。如果取负值，则表示反向偏移，参考原点保持不变。

【示例】本示例将设计向右下角方向平移图像，其中 x 轴偏移 150 像素，y 轴偏移 50 像素，代码如下，演示效果如图 12.4 所示。

```
img.change {
    border: solid 1px red;
    transform: translate(150px, 50px);

}
```

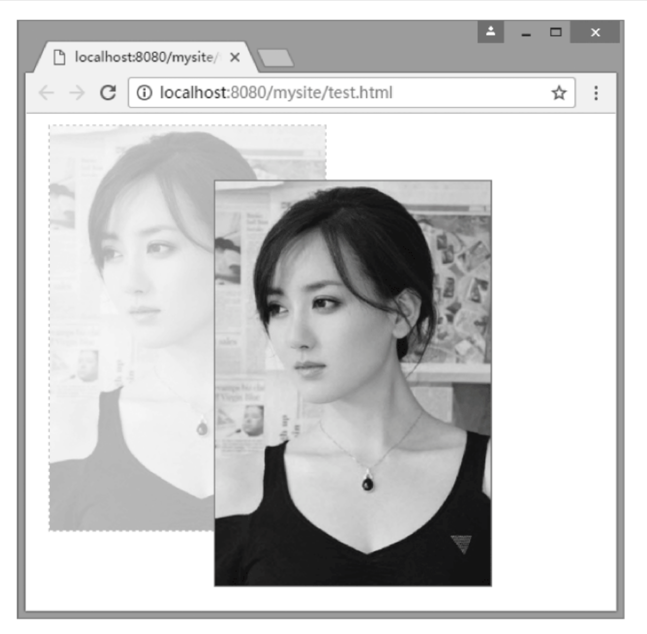


图 12.4　平移对象效果

12.1.6　2D 倾斜



skew()函数能够倾斜显示对象，语法格式如下。

```
skew(<angle> [, <angle>])
```

该函数包含两个参数值，分别用来定义对象在 x 轴和 y 轴倾斜的角度。如果省略参数，则默认值为 0。与 rotate()函数不同，rotate()函数只是旋转对象的角度，而不会改变对象的形状，skew()函数会改变对象的形状。

【示例】本示例将使用 skew()函数变形图像，x 轴倾斜 30°，y 轴倾斜 20°，代码如下，效果如图 12.5 所示。




Note

```
img.change {
    border: solid 1px red;
    transform: skew(30deg, 20deg);
}
```

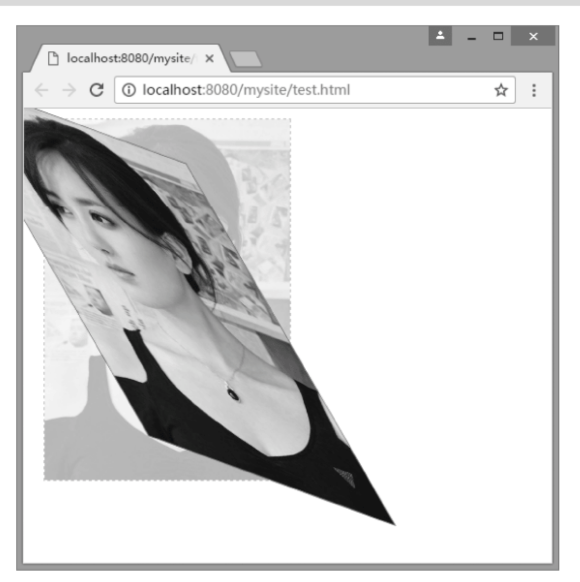


图 12.5　倾斜对象效果


视频讲解

12.1.7　2D 矩阵

matrix()是一个矩阵函数，它可以同时实现缩放、旋转、平移和倾斜操作，语法格式如下。

```
matrix(<number>, <number>, <number>, <number>, <number>, <number>)
```

该函数包含 6 个值，具体说明如下。

☑　第 1 个参数控制 x 轴缩放；

☑　第 2 个参数控制 x 轴倾斜；

☑　第 3 个参数控制 y 轴倾斜；

☑　第 4 个参数控制 y 轴缩放；

☑　第 5 个参数控制 x 轴平移；

☑　第 6 个参数控制 y 轴平移。

【示例】本示例将使用 matrix()函数模拟 12.1.6 节示例的倾斜变形操作，代码如下，效果类似图 12.5 所示效果。

```
img.change {
    border: solid 1px red;
    transform: matrix(1, 0.6, 0.2, 1, 0, 0);
}
```





【补充】

多个变形函数可以在一个声明中同时定义，代码如下。

```
div {
    transform: translate(80, 80);
    transform: rotate(45deg);
    transform: scale(1.5, 1.5);
}
```

针对上面的样式，可以简化为如下代码。

```
div { transform: translate(80, 80) rotate(45deg) scale(1.5, 1.5);}
```

12.1.8　设置变形类型



CSS3 变形包括 2D 和 3D 两种类型，使用 transform-style 属性可以设置 CSS 变形的类型，语法格式如下。

```
transform-style：flat | preserve-3d
```

取值简单说明如下。

☑　flat：指定子元素定位在该元素所在平面内进行变形，即 2D 平面变形，为默认值。

☑　preserve-3d：指定子元素定位在三维空间内进行变形，即 3D 立体变形。

【示例】借助 12.1.7 节示例，本示例使用<div id="box">容器包裹两幅图像，改进后的 HTML 结构如下。

```
<div id="box">
    <img class="bg" src="images/1.jpg">
    <img class="change" src="images/1.jpg">
</div>
```

为<div id="box">容器设置 CSS3 变形类型为 3D，样式代码如下。

```
#box {
    transform-style: preserve-3d;
}
```

为 change 图像应用 3D 顺时针旋转 45°，CSS 样式如下。在浏览器中预览，如图 12.6 所示。

```
img.change {
    border: solid 1px red;
    transform: translate3d(60px, 60px, 400px);
}
```





Note

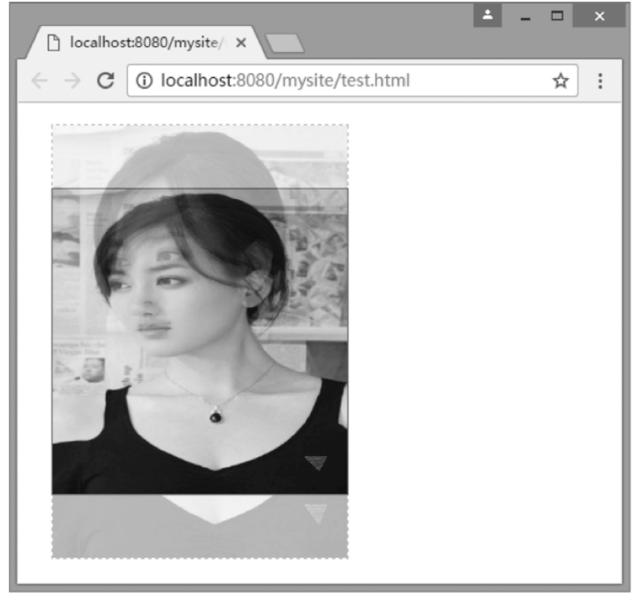


图 12.6　3D 平移效果



视频讲解

12.1.9　设置透视距离和原点

3D 变形与 2D 变形最大的不同就在于其参考的坐标轴不同：2D 变形的坐标轴是平面的，只存在 x 轴和 y 轴，而 3D 变形的坐标轴则是 x、y、z 这 3 条轴组成的立体空间，x 轴正向、y 轴正向、z 轴正向分别朝向右、下和屏幕外，示意图如图 12.7 所示。

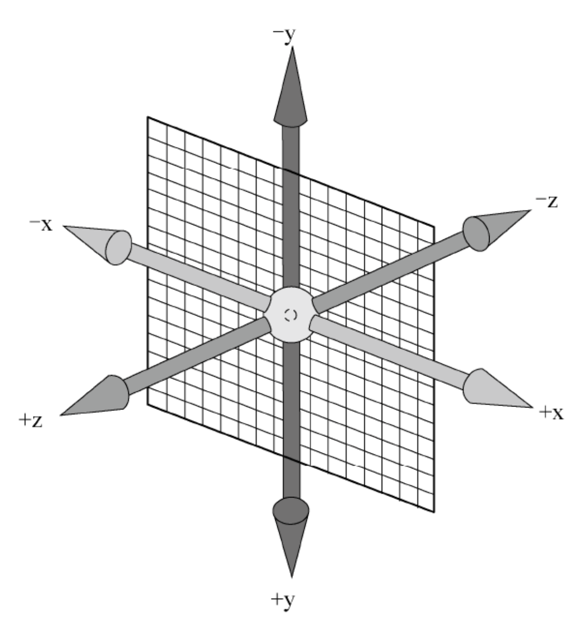


图 12.7　3D 坐标轴示意图





透视是 3D 变形中最重要的概念。如果不设置透视，元素的 3D 变形效果将无法实现。在上节示例中，使用函数 rotateX(45deg)将图像以 x 轴方向为轴沿顺时针旋转 45°，由于没有设置透视样式的效果，可以看到浏览器将图像的 3D 变形操作垂直投射到 2D 视平面上，最终呈现出来的只是图像的宽高变化。

【示例 1】在上节示例基础上，在<div id="box">容器外设置透视点距离为 1200 像素，样式代码如下。

```
body{
    perspective: 1200px;
}
```

在浏览器中可以看到如图 12.8 所示的变形效果。

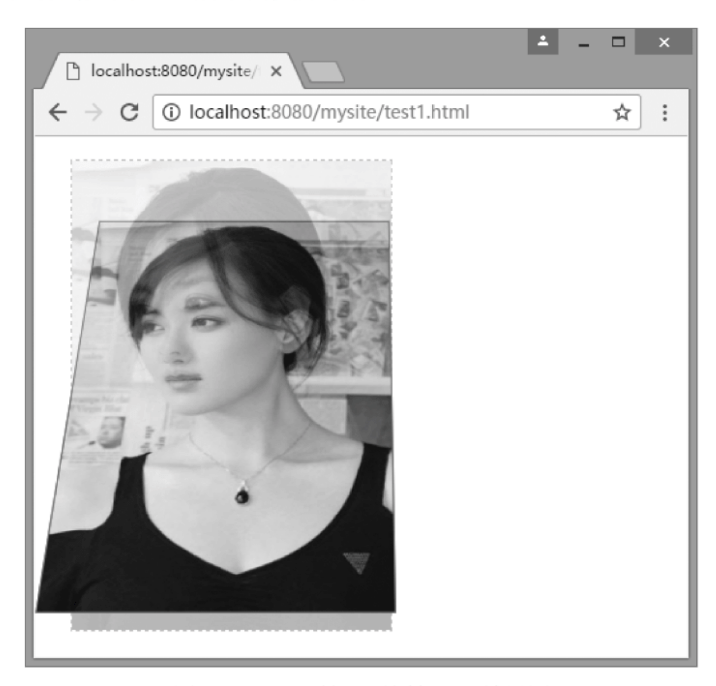


图 12.8　沿 X 轴 3D 旋转 45°效果图

基于对上面示例的直观体验，下面来了解几个核心概念：变形元素、观察者和被透视元素，关系如图 12.9 所示。

- ☑ 变形元素：需要进行 3D 变形的元素。主要用来设置 transform、transform-origin、backface-visibility 等属性。
- ☑ 观察者：浏览器模拟出来的用于观察被透视元素的一个没有尺寸的点，观察者发出视线，类似于一个点光源发出光线。
- ☑ 被透视元素：被观察者观察的元素，根据属性设置不同，它有可能是变形对象本身，也可能是它的父级或祖先元素，主要用来设置 perspective、perspective-origin 等属性。





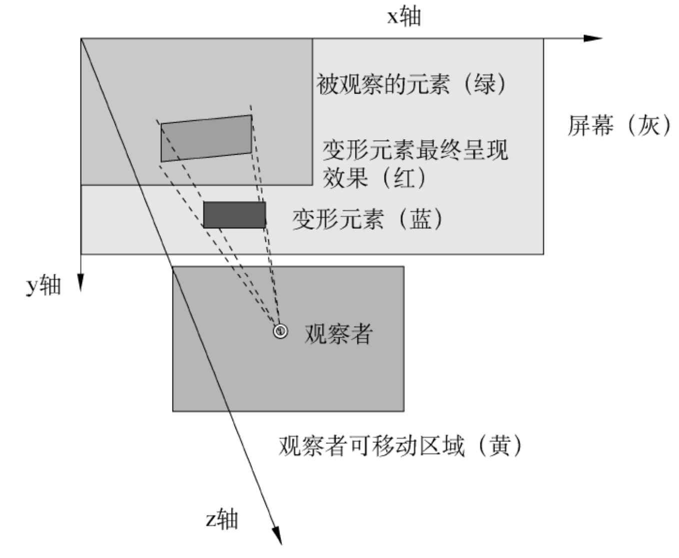


图 12.9　变形元素、观察者和被透视元素位置关系示意图

1．透视距离

透视距离是指观察者沿着平行于 z 轴的视线与屏幕之间的距离，如图 12.10 所示。

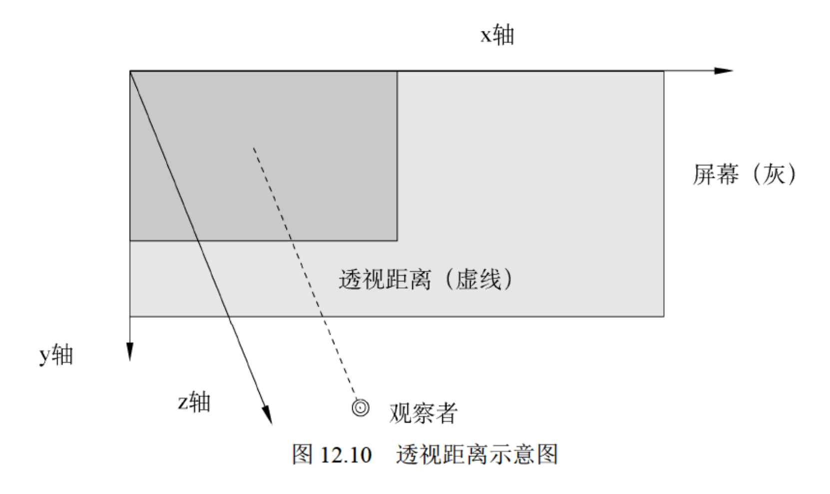


图 12.10　透视距离示意图

使用 perspective 属性可以定义透视距离，语法格式如下。

```
perspective：none | <length>
```

取值简单说明如下。

☑　none：不指定透视。

☑　<length>：指定观察者距离平面的距离，为元素及其内容应用透视变换。

注意，透视距离不可为 0 和负数，因为观察者与屏幕距离为 0 时或者在屏幕背面时是不可以观察到被透视元素的正面的。perspective 也不可取百分比，因为百分比需要相对的元素，但 z 轴并没有



可相对的元素尺寸。

一般地，物体离得越远，显得越小。反映在 perspective 属性上，就是该属性值越大，元素的 3D 变形效果越不明显。

设置 perspective 属性的元素就是被透视元素。一般地，该属性只能设置在变形元素的父级或祖先级。因为浏览器会为其子级的变形产生透视效果，但并不会为其自身产生透视效果。应用示例可以参考本节示例 1。

2. 透视原点

透视原点是指观察者的位置，一般观察者位于与屏幕平行的另一个平面上，观察者始终是与屏幕垂直的。观察者的活动区域是被观察元素的盒模型区域，示意图如图 12.11 所示。

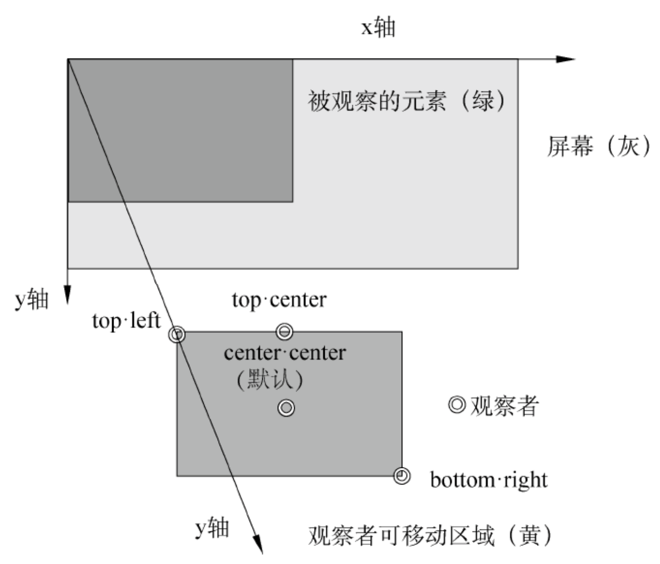


图 12.11　下面方形区域为透视原点的位置区域

使用 perspective-origin 属性可以定义透视点的位置，语法格式如下。

```
perspective-origin: [ <percentage> | <length> | left | center① | right ] [ <percentage> | <length> | top | center② | bottom ]?
```

取值简单说明如下。

☑　<percentage>：用百分比指定透视点坐标值，相对于元素宽度。可以为负值。

☑　<length>：用长度值指定透视点坐标值。可以为负值。

☑　left：指定透视点的横坐标为 left。

☑　center①：指定透视点的横坐标为 center。

☑　right：指定透视点的横坐标为 right。

☑　top：指定透视点的纵坐标为 top。

☑　center②：指定透视点的纵坐标为 center。

☑　bottom：指定透视点的纵坐标为 bottom。





Note

【示例 2】在示例 1 的基础上，设置观察点位置在右侧居中位置。代码如下，显示效果如图 12.12 所示。

```
body{
    perspective: 1200px;
    perspective-origin: right;
}
```

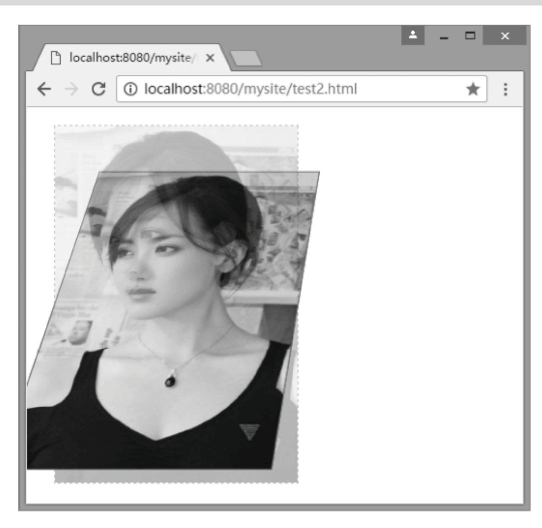


图 12.12　设置观察点位置在右侧的效果



视 频 讲 解

12.1.10　3D 平移

3D 平移主要包括下面 4 个函数。

☑ translatex(<translation-value>)：指定对象 x 轴（水平方向）的平移。

☑ translatey(<translation-value>)：指定对象 y 轴（垂直方向）的平移。

☑ translatez(<length>)：指定对象 z 轴的平移。

☑ translate3d(<translation-value>,<translation-value>,<length>)：指定对象的 3D 平移。第 1 个参数对应 x 轴，第 2 个参数对应 y 轴，第 3 个参数对应 z 轴，参数不允许省略。

参数<translation-value>表示<length>或<percentage>，即 x 轴和 y 轴可以取长度值或百分比，但是 z 轴只能够设置长度值。

【示例】本示例设计图像在 3D 空间中平移，使其呈现一种错位效果。代码如下，效果如图 12.13 所示。

```
#box {
    transform-style: preserve-3d;
    perspective: 1200px;
}
img.change {
```



```
        border: solid 1px red;
        transform: translate3d(200px, 30px, 60px);
    }
```



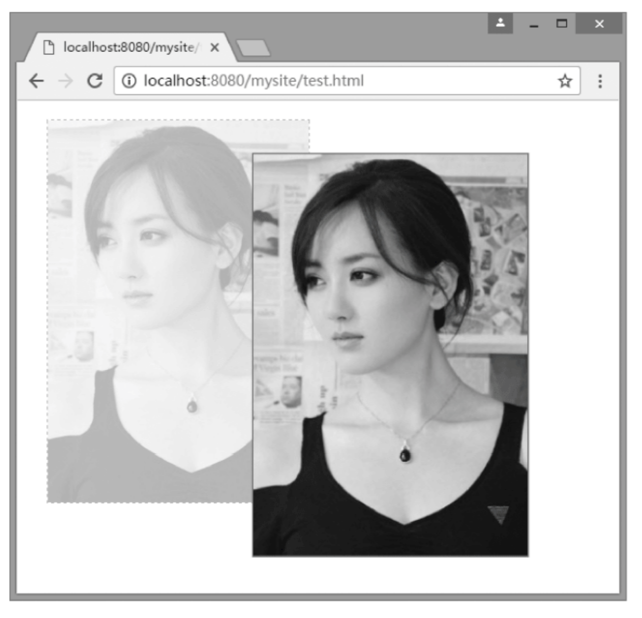


图 12.13　定义 3D 平移效果

从图 12.13 可以看出，当 z 轴值越大时，元素离浏览者越近，视觉上元素就变得更大；反之其值越小时，元素也离观看者越远，视觉上元素就变得更小。

> **提示：** translatez()函数在实际使用中等效于 translate3d(0,0,tz)。仅从视觉效果上看，translatez()和 translate3d(0,0,tz)函数功能非常类似于二维空间的 scale()缩放函数，但实际上完全不同。translatez()和 translate3d(0,0,tz)变形是发生在 z 轴上，而不是 x 轴和 y 轴。

12.1.11　3D 缩放



3D 缩放主要包括以下 4 个函数。

☑　scalex(<number>)：指定对象 x 轴的（水平方向）缩放。

☑　scaley(<number>)：指定对象 y 轴的（垂直方向）缩放。

☑　scalez(<number>)：指定对象的 z 轴缩放。

☑　scale3d(<number>,<number>,<number>)：指定对象的 3D 缩放。第 1 个参数对应 x 轴，第 2 个参数对应 y 轴，第 3 个参数对应 z 轴，参数不允许省略。

参数<number>为一个数字，表示缩放倍数，可参考 2D 缩放参数说明。

【示例】以上面示例为基础，在 x 轴和 y 轴上放大图像 1.5 倍，z 轴上放大图像两倍，然后使用 translatex()把变形的图像移到右侧显示，以便与原图进行比较。代码如下，演示效果如图 12.14 所示。

```
img.change {
```





Note

```
        border: solid 1px red;
        transform: scale3D(1.5,1.5,2) translatex(240px);
    }
```

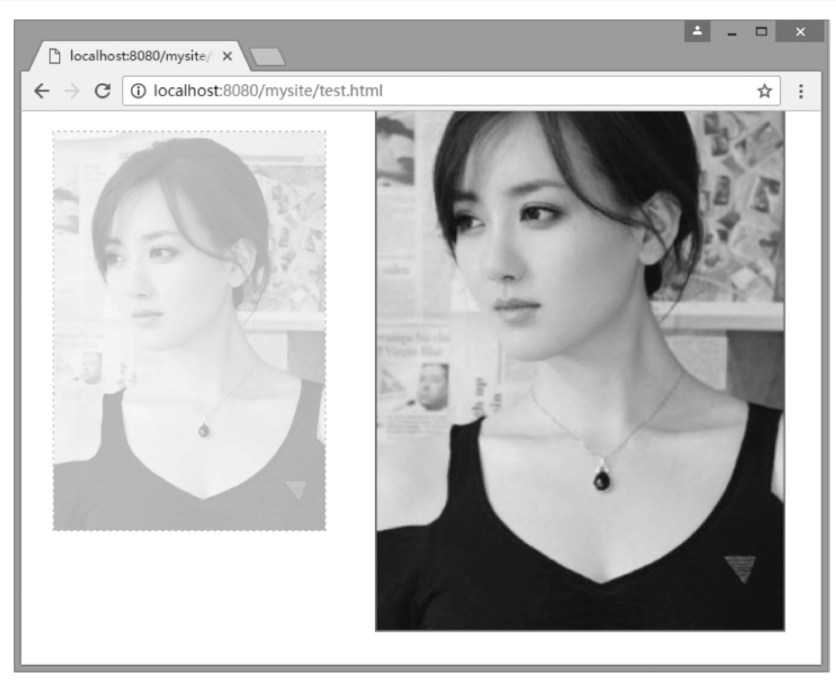


图 12.14　定义 3D 缩放效果



视频讲解

12.1.12　3D 旋转

3D 旋转主要包括下面 4 个函数。

☑　rotatex(<angle>)：指定对象在 x 轴上的旋转角度。

☑　rotatey(<angle>)：指定对象在 y 轴上的旋转角度。

☑　rotatez(<angle>)：指定对象在 z 轴上的旋转角度。

☑　rotate3d(<number>,<number>,<number>,<angle>)：指定对象的 3D 旋转角度，其中前 3 个参数分别表示旋转的方向 x、y、z，第 4 个参数表示旋转的角度，参数不允许省略。

提示： rotate3d()函数前 3 个参数值分别用来描述围绕 x、y、z 轴旋转的矢量值。最终变形元素以由(0,0,0)和(x,y,z)这两个点构成的直线为轴进行旋转。当第 4 个参数为正数时，元素进行顺时针旋转；当第 4 个参数为负数时，元素进行逆时针旋转。

rotate3d()函数可以与前面 3 个旋转函数进行转换，简单说明如下。

☑　rotatex(a)函数功能等同于 rotate3d(1,0,0,a)。

☑　rotatey(a)函数功能等同于 rotate3d(0,1,0,a)。

☑　rotatez(a)函数功能等同于 rotate3d(0,0,1,a)。

【示例】 以 12.1.11 节示例为基础，使用 rotate3d()函数顺时针旋转图像 45°，其中 x 轴、y 轴和





z 轴比值为 2、2、1。代码如下，效果如图 12.15 所示。

```
img.change {
    border: solid 1px red;
    transform: rotate3d(2,2,1,45deg);
}
```

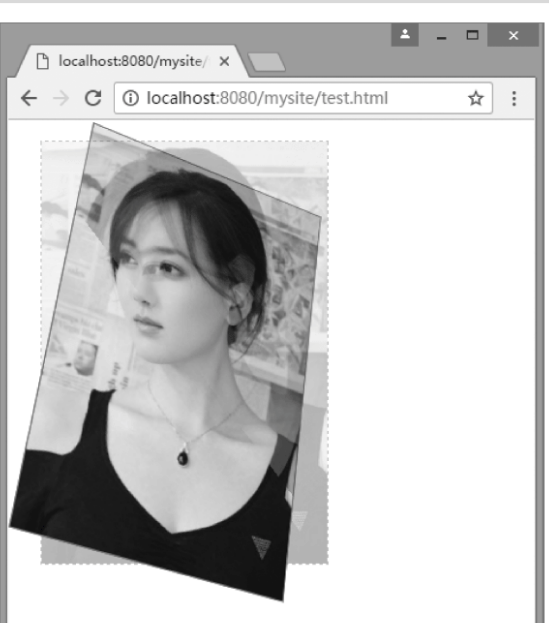


图 12.15　定义 3D 旋转效果

12.1.13　透视函数



perspective 属性可以定义透视距离，它必须应用在变形元素的父级或祖先级元素上。而透视函数 perspective()是 transform 变形函数的一个属性值，可以应用于变形元素本身。具体语法格式如下。

```
perspective(<length>)
```

参数是一个长度值，值只能是正数。

【示例】本示例设计图像在 x 轴上旋转 120°，透视距离为 1200 像素。代码如下，效果如图 12.16 所示。

```
#box { transform-style: preserve-3d;}
img.change {
    border: solid 1px red;
    transform:perspective(180px) rotateX(120deg);
}
```




Note

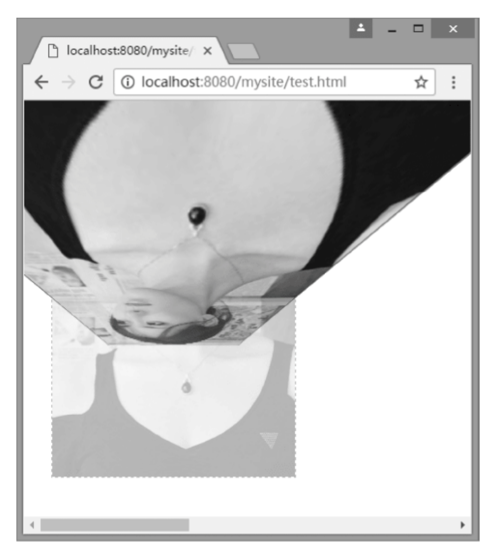


图 12.16　定义 3D 旋转效果

注意：由于 transform 属性是从前向后的顺序解析属性值的，所以一定要把 perspective()函数写在其他变形函数前面，否则将没有透视效果。

perspective-origin 只能设置在设置了 perspective 透视属性的元素上。若为元素设置透视函数 perspective()，则透视原点不起作用。观察者使用默认位置，即元素中心点对应的平面上。

12.1.14　变形原点

2D 变形原点由于没有 z 轴，所以 z 轴的值默认为 0。在 3D 变形原点中，z 轴是一个可以设置的变量。语法格式如下。

```
transform-origin: x 轴  y 轴  z 轴
```

取值简单说明如下。

☑　x 轴: left | center | right | <length> | <percentage>。

☑　y 轴: top | center | bottom | <length> | <percentage>。

☑　z 轴: <length>。

对于 x 轴和 y 轴来说，可以设置关键字和百分比，分别相对于其本身元素水平方向的宽度和垂直方向的高度，而 z 轴只能设置长度值。


视频讲解

12.1.15　背景可见

元素的背面在默认情况下是可见的，有时可能需要让元素背面不可见，这时候就可以使用 backface-visibility 属性，该属性的具体语法格式如下。

```
backface-visibility：visible | hidden
```





取值简单说明如下。

☑　visible：指定元素背面可见，允许显示正面的镜像，为默认值。

☑　hidden：指定元素背面不可见。

【示例】以 12.1.13 节的示例为基础，在变形图像样式中添加 backface-visibility: hidden;，定义元素背面面向用户时不可见，这时再次预览，会发现变形图像已经不存在，因为它的背面面向用户，被隐藏了。代码如下，效果如图 12.17 所示。

```
img.change {
    border: solid 1px red;
    transform:perspective(180px) rotateX(120deg);
    backface-visibility: hidden;
}
```

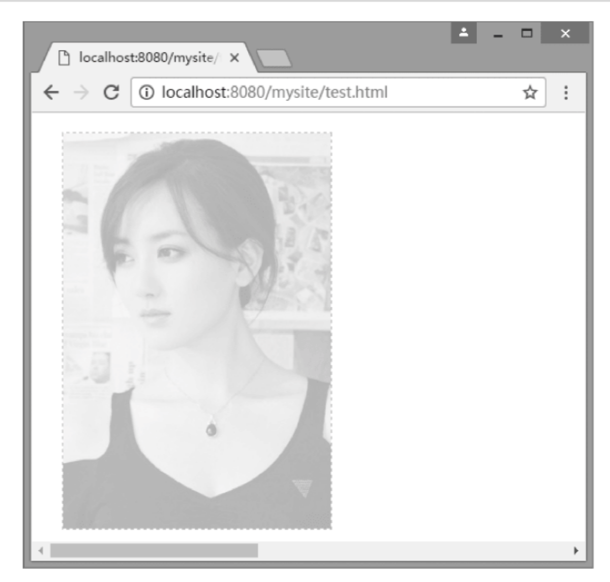


图 12.17　定义背面面向用户不可见效果

12.2　过渡动画

2013 年 2 月，W3C 发布了 CSS Transitions 工作草案，在这个草案中描述了 CSS 过渡动画的基本实现方法和属性。目前获得了所有浏览器的支持，包括支持带有前缀（私有属性）或不带前缀的过渡（标准属性）。最新版本浏览器（IE 10+、Firefox 16+和 Opera 12.5+）均支持不带前缀的过渡属性 transition，而旧版浏览器则支持带有前缀的过渡属性，如 Webkit 引擎支持-webkit-transition 私有属性，Mozilla Gecko 引擎支持-moz-transition 私有属性，Presto 引擎支持-o-transition 私有属性，IE 6～IE 9 浏览器不支持 transition 属性，IE10 支持 transition 属性。

权威参考：http://www.w3.org/TR/css3-transitions/。



12.2.1　设置过渡属性





transition-property 属性用来定义过渡动画的 CSS 属性名称，基本语法如下。

```
transition-property:none | all | [ <IDENT> ] [ ',' <IDENT> ]*;
```

取值简单说明如下。

☑　none：表示没有元素。

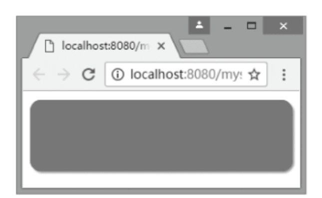
☑　all：默认值，表示针对所有元素，包括:before 和:after 伪元素。

☑　IDENT：指定 CSS 属性列表。几乎所有色彩、大小或位置等相关的 CSS 属性，包括许多新添加的 CSS3 属性，都可以应用过渡，如 CSS3 变换中的放大、缩小、旋转、斜切、渐变等。

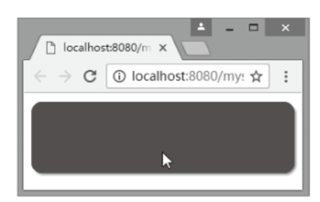
【示例】在本示例中，指定动画的属性为背景颜色。这样当鼠标指针经过盒子时，会自动从红色背景过渡到蓝色背景。代码如下，演示效果如图 12.18 所示。

```
<style type="text/css">
div {
    margin: 10px auto; height: 80px;
    background: red;
    border-radius: 12px;
    box-shadow: 2px 2px 2px #999;
}
div:hover {
    background-color: blue;
    /*指定动画过渡的 CSS 属性*/
    transition-property: background-color;
}
</style>

<div></div>
```



默认状态



鼠标指针经过时被旋转

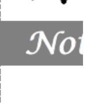
图 12.18　定义简单的背景色切换动画



12.2.2　设置过渡时间

transition-duration 属性用来定义转换动画的时间长度，基本语法如下。





```
transition-duration:<time> [, <time>]*;
```

初始值为 0，适用于所有元素，以及:before 和:after 伪元素。在默认情况下，动画过渡时间为 0s，所以当指定元素动画时，会看不到过渡的过程，直接看到结果。

【示例】以 12.2.1 节示例为基础，本示例设置动画过渡时间为 2s，当鼠标指针移过对象时，会看到背景色从红色逐渐过渡到蓝色。代码如下，演示效果如图 12.19 所示。

```
div:hover {
    background-color: blue;
    /*指定动画过渡的 CSS 属性*/
    transition-property: background-color;
    /*指定动画过渡的时间*/
    transition-duration:2s;
}
```

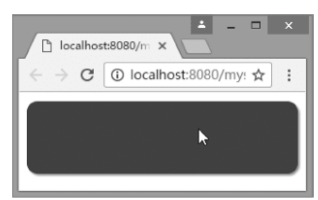


图 12.19　设置动画时间

12.2.3　设置延迟过渡时间



transition-delay 属性用来定义开启过渡动画的延迟时间，基本语法如下。

```
transition-delay:<time> [, <time>]*;
```

初始值为 0，适用于所有元素，以及:before 和:after 伪元素。设置时间可以为正整数、负整数和 0，非零的时候必须设置单位是 s（秒）或者 ms（毫秒）；为负数的时候，过渡的动作会从该时间点开始显示，之前的动作被截断；为正数的时候，过渡的动作会延迟触发。

【示例】继续以 12.2.1 节示例为基础进行介绍，本示例设置过渡动画推迟 2s 后执行，则当鼠标指针移过对象时，会看不到任何变化，过了 2s 之后，才发现背景色从红色逐渐过渡到蓝色，代码如下。

```
div:hover {
    background-color: blue;
    /*指定动画过渡的 CSS 属性*/
    transition-property: background-color;
    /*指定动画过渡的时间*/
    transition-duration: 2s;
    /*指定动画延迟触发 */
```



```
    transition-delay: 2s;
}
```



Note



视频讲解

12.2.4 设置过渡动画类型

transition-timing-function 属性用来定义过渡动画的类型，基本语法如下。

```
transition-timing-function:ease | linear | ease-in | ease-out | ease-in-out | cubicbezier(<number>, <number>, <number>, <number>) [, ease | linear | ease-in | ease-out | ease-in-out | cubic-bezier(<number>, <number>,<number>, <number>)]*
```

属性初始值为 ease，取值简单说明如下。

☑ ease：平滑过渡，等同于 cubic-bezier(0.25, 0.1, 0.25, 1.0)函数，即立方贝塞尔。
☑ linear：线性过渡，等同于 cubic-bezier(0.0, 0.0, 1.0, 1.0)函数。
☑ ease-in：由慢到快，等同于 cubic-bezier(0.42, 0, 1.0, 1.0)函数。
☑ ease-out：由快到慢，等同于 cubic-bezier(0, 0, 0.58, 1.0)函数。
☑ ease-in-out：由慢到快再到慢，等同于 cubic-bezier(0.42, 0, 0.58, 1.0)函数。
☑ cubic-bezier：特殊的立方贝塞尔曲线效果。

【示例】继续以 12.2.1 节示例为基础进行介绍，本示例设置过渡类型为线性，代码如下。

```
div:hover {
    background-color: blue;
    /*指定动画过渡的 CSS 属性*/
    transition-property: background-color;
    /*指定动画过渡的时间*/
    transition-duration: 10s;
    /*指定动画过渡为线性效果 */
    transition-timing-function: linear;
}
```



视频讲解

12.2.5 设置过渡触发动作

CSS3 过渡动画一般通过动态伪类触发，如表 12.1 所示。

表 12.1 CSS 动态伪类

动 态 伪 类	作 用 元 素	说　　明
:link	只有链接	未访问的链接
:visited	只有链接	访问过的链接
:hover	所有元素	鼠标指针经过元素
:active	所有元素	鼠标单击元素
:focus	所有可被选中的元素	元素被选中



也可以通过 JavaScript 事件触发，包括 click、focus、mousemove、mouseover、mouseout 等。



1. :hover

最常用的过渡触发方式是使用:hover 伪类。

【示例 1】本示例设计当鼠标指针经过 div 元素时，该元素的背景颜色会在经过 1s 的初始延迟后，于 2s 内动态地从绿色变为蓝色，代码如下。

```
<style type="text/css">
div {
    margin: 10px auto;
    height: 80px;
    border-radius: 12px;
    box-shadow: 2px 2px 2px #999;
    background-color: red;
    transition: background-color 2s ease-in 1s;
}
div:hover { background-color: blue}
</style>

<div></div>
```

2. :active

:active 伪类表示用户单击某个元素并按住鼠标按键时显示的状态。

【示例 2】本示例设计当用户单击 div 元素时，该元素被激活，这时会触发动画，高度属性从 200px 过渡到 400px。如果按住该元素，保持住活动状态，则 div 元素始终显示 400px 高度，松开鼠标之后，又会恢复原来的高度。代码如下，效果如图 12.20 所示。

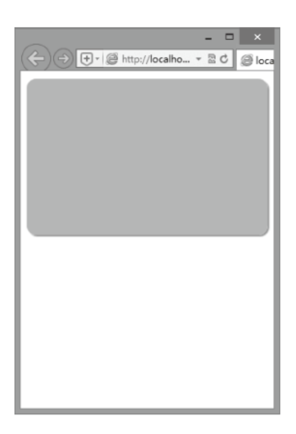
```
<style type="text/css">
div {
    margin: 10px auto;
    border-radius: 12px;
    box-shadow: 2px 2px 2px #999;
    background-color: #8AF435;
    height: 200px;
    transition: width 2s ease-in;
}
div:active {height: 400px;}
</style>

<div></div>
```

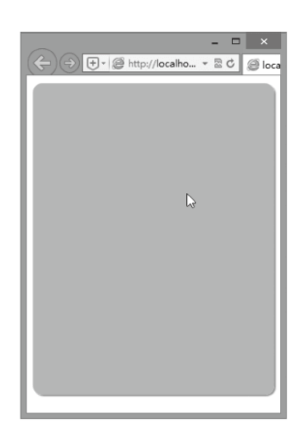








默认状态



单击

图 12.20　定义激活触发动画

3. :focus

:focus 伪类通常会在表单对象接收键盘响应时出现。

【示例 3】本示例将设计当输入框获取焦点时，输入框的背景色逐步高亮显示。代码如下，效果如图 12.21 所示。

```
<style type="text/css">
label {
    display: block;
    margin: 6px 2px;
}
input[type="text"], input[type="password"] {
    padding: 4px;
    border: solid 1px #ddd;
    transition: background-color 1s ease-in;
}
input:focus { background-color: #9FFC54;}
</style>

<form id=fm-form action="" method=post>
    <fieldset>
        <legend>用户登录</legend>
        <label for="name">姓名
            <input type="text" id="name" name="name" >
        </label>
        <label for="pass">密码
            <input type="password" id="pass" name="pass" >
```





```
            </label>
        </fieldset>
    </form>
```





图 12.21　定义获取焦点触发动画

提示：把:hover 伪类与:focus 配合使用，能够丰富鼠标用户和键盘用户的体验。

4. :checked

:checked 伪类在发生选中状况时触发过渡。

【示例 4】本示例设计当复选框被选中时缓慢缩进两个字符。代码如下，演示效果如图 12.22 所示。

```
<style type="text/css">
label.name {
    display: block;
    margin: 6px 2px;
}
input[type="text"], input[type="password"] {
    padding: 4px;
    border: solid 1px #ddd;
}
input[type="checkbox"] { transition: margin 1s ease;}
input[type="checkbox"]:checked { margin-left: 2em;}
</style>

<form id=fm-form action="" method=post>
    <fieldset>
        <legend>用户登录</legend>
        <label class="name" for="name">姓名
            <input type="text" id="name" name="name" >
        </label>
        <p>技术专长<br>
            <label>
                <input type="checkbox" name="web" value="html" id="web_0">
```





Note

```
                    HTML</label><br>
                <label>
                    <input type="checkbox" name="web" value="css" id="web_1">
                    CSS</label><br>
                <label>
                    <input type="checkbox" name="web" value="javascript" id="web_2">
                    JavaScript</label><br>
            </p>
        </fieldset>
    </form>
```

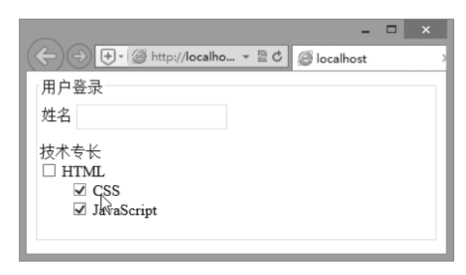


图 12.22　定义被选中时触发动画

5. 媒体查询

触发元素状态变化的另一种方法是使用 CSS3 媒体查询。

【示例 5】 本示例设计 div 元素的宽度和高度为 49%×200px，如果用户将窗口大小调整到 420px 或以下，则该元素将过渡为 100%×100px。也就是说，当窗口宽度变化经过 420px 的阈值时，将会触发过渡动画。代码如下，演示效果如图 12.23 所示。

```
<style type="text/css">
div {
    float: left; margin: 2px;
    width: 49%; height: 200px;
    background: #93FB40;
    border-radius: 12px;
    box-shadow: 2px 2px 2px #999;
    transition: width 1s ease, height 1s ease;
}
@media only screen and (max-width : 420px) {
    div {
        width: 100%;
        height: 100px;
    }
```

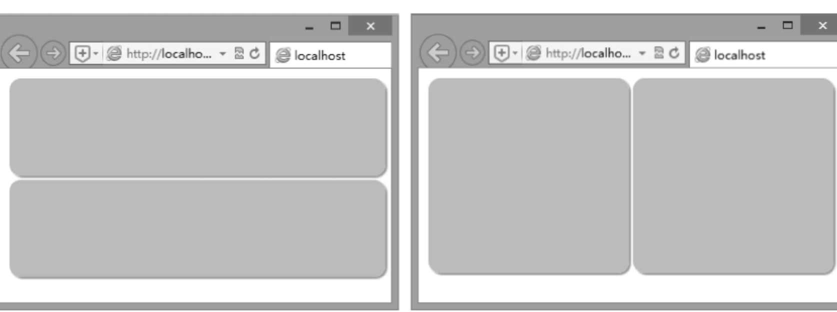






```
}
</style>

<div></div>
<div></div>
```



当窗口小于等于420px宽度　　　　当窗口大于420px宽度

图 12.23　设备类型触发动画

如果网页加载时用户的窗口大小是 420px 或以下，浏览器会在该部分应用这些样式，但是由于不会出现状态变化，因此不会发生过渡。

6. JavaScript 事件

【示例 6】本示例可以使用纯粹的 CSS 伪类触发过渡，为了方便用户理解，这里通过 jQuery 脚本触发过渡，代码如下。

```
<script type="text/javascript" src="images/jquery-1.10.2.js"></script>
<script type="text/javascript">
$(function() {
    $("#button").click(function() {
        $(".box").toggleClass("change");
    });
});
</script>
<style type="text/css">
.box {
    margin:4px;
    background: #93FB40;
    border-radius: 12px;
    box-shadow: 2px 2px 2px #999;
    width: 50%; height: 100px;
    transition: width 2s ease, height 2s ease;
```

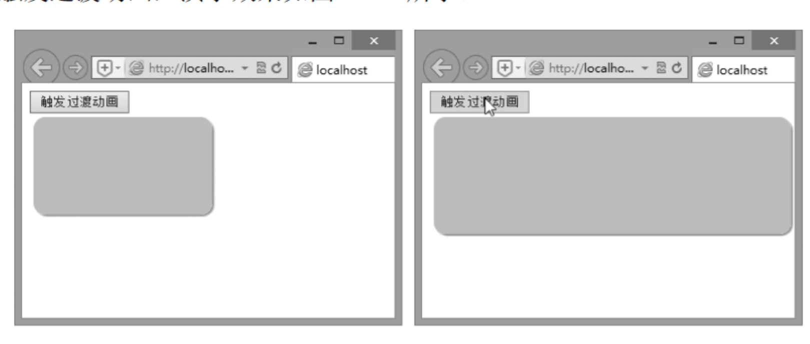




```
}
.change {
    width: 100%;
    height: 120px;
}
</style>

<input type="button" id="button" value="触发过渡动画" />
<div class="box"></div>
```

文档中包含一个 box 类的盒子和一个按钮，当单击按钮时，jQuery 脚本会将盒子的类切换为 change，从而触发过渡动画，演示效果如图 12.24 所示。



默认状态　　　　JavaScript事件激活状态

图 12.24　使用 JavaScript 脚本触发动画

上面示例演示了样式发生变化会导致过渡动画，也可以通过其他方法触发这些变化，包括通过 JavaScript 脚本动态更改。从执行效率来看，事件通常应当通过 JavaScript 触发，简单动画或过渡则应使用 CSS 触发。当然，这只是一般指导原则，不一定是最佳选择，具体应视条件而定。

12.3　帧　动　画

2012 年 4 月，W3C 发布了 CSS Animations 工作草案，在这个草案中描述了 CSS 关键帧动画的基本实现方法和属性。目前最新版本的主流浏览器都支持 CSS 帧动画，如 IE 10+、Firefox 和 Opera 均支持不带前缀的动画属性 animation，而旧版浏览器则支持前缀的动画，如 Webkit 引擎支持-webkit-animation 属性，Mozilla Gecko 引擎支持-moz-animation 私有属性，Presto 引擎支持-o-animation 私有属性，IE 6～IE 9 浏览器不支持 animation 属性。

权威参考：http://www.w3.org/TR/css3-animations/。


视频讲解

12.3.1　设置关键帧

CSS3 使用@keyframes 定义关键帧，具体用法如下。





```
@keyframes animationname {
    keyframes-selector {
        css-styles;
    }
}
```

其中参数说明如下。

☑ animationname：定义动画的名称。

☑ keyframes-selector：定义帧的时间未知，也就是动画时长的百分比，合法的值包括 0～100%、from（等价于 0）、to（等价于 100%）。

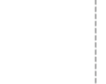
☑ css-styles：表示一个或多个合法的 CSS 样式属性。

在动画过程中，用户能够多次改变这套 CSS 样式。以百分比或者通过关键词 from 和 to 来定义样式改变发生的时间。为了获得最佳浏览器支持，设计关键帧动画时，应该始终定义 0 和 100%位置帧。最后，为每帧定义动态样式，同时将动画与选择器绑定。

【示例】 本示例演示了如何让一个小方盒沿着方形框内壁匀速运动。代码如下，效果如图 12.25 所示。

```
<style>
#wrap {/* 定义运动轨迹包含框*/
    position:relative;                /* 定义定位包含框，避免小盒子跑到外面运动*/
    border:solid 1px red;
    width:250px; height:250px;
}
#box {/* 定义运动小盒的样式*/
    position:absolute;
    left:0; top:0;
    width: 50px; height: 50px;
    background: #93FB40;
    border-radius: 8px;
    box-shadow: 2px 2px 2px #999;
/*定义帧动画：动画名称为 ball，动画时长 5s，动画类型为匀速渐变，动画无限播放*/
    animation: ball 5s linear infinite;
}
/*定义关键帧：共包括 5 帧，分别在总时长 0%、25%、50%、75%、100%的位置*/
/*在每帧中设置动画属性为 left 和 top，让它们的值匀速渐变，产生运动动画*/
@keyframes ball {
    0% {left:0;top:0;}
    25% {left:200px;top:0;}
    50% {left:200px;top:200px;}
```





```
        75% {left:0;top:200px;}
        100% {left:0;top:0;}
    }
    </style>

    <div id="wrap">
        <div id="box"></div>
    </div>
```

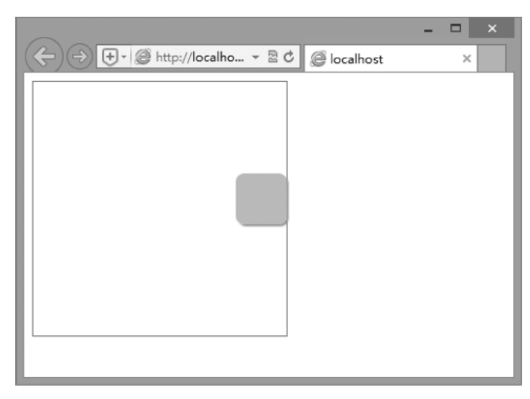


图 12.25　设计小盒子运动动画



视频讲解

12.3.2　设置动画属性

Animations 功能与 Transition 功能相同，都是通过改变元素的属性值来实现动画效果的。它们的区别在于：使用 Transitions 功能时只能通过指定属性的开始值与结束值，然后在这两个属性值之间进行平滑过渡来实现动画效果，因此不能实现比较复杂的动画效果；而 Animations 则通过定义多个关键帧和每个关键帧中元素的属性值来实现更为复杂的动画效果。

1. 定义动画名称

使用 animation-name 属性可以定义 CSS 动画的名称，语法如下。

```
animation-name:none | IDENT [, none | IDENT ]*;
```

初始值为 none，定义一个适用的动画列表。每个名字是用来选择动画关键帧的，以提供动画的属性值。如名称是 none，那么就不会有动画。

2. 定义动画时间

使用 animation-duration 属性可以定义 CSS 动画的播放时间，语法如下。

```
animation-duration:<time> [, <time>]*;
```

在默认情况下该属性值为 0，这意味着动画周期是 0，即不会有动画。当值为负值时，则被视为 0。





3. 定义动画类型

使用 animation-timing-function 属性可以定义 CSS 动画类型，语法如下。

```
animation-timing-function:ease | linear | ease-in | ease-out | ease-in-out | cubicbezier(<number>, <number>, number>, <number>) [, ease | linear |ease-in | ease-out | ease-in-out | cubic-bezier(<number>, <number>,<number>, <number>)]*
```

初始值为 ease，取值说明可参考 12.2.4 节。

4. 定义延迟时间

使用 animation-delay 属性可以定义 CSS 动画延迟播放的时间，语法如下。

```
animation-delay:<time> [, <time>]*;
```

该属性允许一个动画开始执行一段时间后才被应用。当动画延迟时间为 0，即默认动画延迟时间，则意味着动画将尽快执行，否则该值指定将延迟执行的时间。

5. 定义播放次数

使用 animation-iteration-count 属性可以定义 CSS 动画的播放次数，语法如下。

```
animation-iteration-count:infinite | <number> [, infinite | <number>]*;
```

默认值为 1，这意味着动画将从开始到结束播放一次。infinite 表示无限次，即 CSS 动画永远重复。如果取值为非整数，将导致动画结束一个周期的一部分。如果取值为负值，则将导致在交替周期内反向播放动画。

6. 定义播放方向

使用 animation-direction 属性可以定义 CSS 动画的播放方向，基本语法如下。

```
animation-direction:normal | alternate [, normal | alternate]*;
```

默认值为 normal。当为默认值时，动画的每次循环都向前播放。另一个值是 alternate，设置该值则表示第偶数次向前播放，第奇数次向反方向播放。

7. 定义播放状态

使用 animation-play-state 属性可以定义动画正在运行还是暂停，语法如下。

```
animation-play-state: paused|running;
```

初始值为 running。其中 paused 定义动画已暂停，running 定义动画正在播放。

提示： 可以在 JavaScript 中使用该属性，这样就能在播放过程中暂停动画。在 JavaScript 脚本中的用法如下。

```
object.style.animationPlayState="paused"
```




Note

8. 定义播放外状态

使用 animation-fill-mode 属性可以定义动画外状态，语法如下。

```
animation-fill-mode：none | forwards | backwards | both [ , none | forwards | backwards | both ]*
```

初始值为 none，如果提供多个属性值，以逗号进行分隔。取值说明如下。

☑ none：不设置对象动画之外的状态。

☑ forwards：设置对象状态为动画结束时的状态。

☑ backwards：设置对象状态为动画开始时的状态。

☑ both：设置对象状态为动画结束或开始时的状态。

【示例】本示例将设计一个小球，并定义它水平向左运动，动画结束之后，再返回起始点位置。代码如下，效果如图 12.26 所示。

```
<style>
/*启动运动的小球，并定义动画结束后返回*/
.ball{
    width: 50px; height: 50px;
    background: #93FB40;
    border-radius: 100%;
    box-shadow:2px 2px 2px #999;
    animation:ball 1s ease backwards;
}
/*定义小球水平运动关键帧*/
@keyframes ball{
    0%{transform:translate(0,0);}
    100%{transform:translate(400px);}
}
</style>

<div class="ball"></div>
```

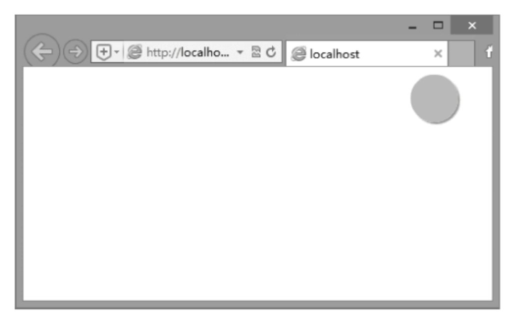


图 12.26　设计运动小球最后返回起始点位置



12.4　案例实战





本节将通过多个案例帮助读者上机练习和提升 CSS3 动画的设计技法。

12.4.1　设计图形

【示例 1】设计菱形。制作菱形的方法有很多种，本例使用 transform 属性和 rotate()函数相结合的方法使两个正反三角形上下显示。代码如下，效果如图 12.27 所示。

```
#shape {
    width: 120px; height: 120px;
    background: #1eff00;
    margin: 60px auto;
    transform: rotate(-45deg);      /* 逆时针旋转 45°  */
    transform-origin: 0 100%;       /* 以右上角为原点进行旋转 */
}
</style>

<div id="shape"></div>
```

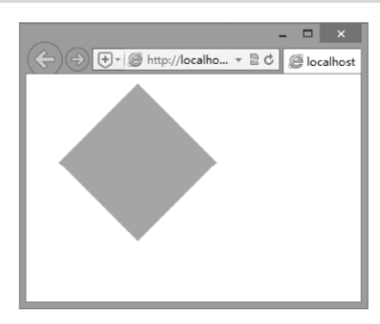


图 12.27　设计菱形

【示例 2】设计平行四边形。制作平行四边形的方法：使用 transform 属性让长方形倾斜一个角度。代码如下，效果如图 12.28 所示。

```
<style type="text/css">
#shape {
    width: 200px; height: 120px;
    background: #1eff00;
    margin: 60px auto;
    transform: skew(30deg);
}
```





Note

```
</style>

<div id="shape"></div>
```

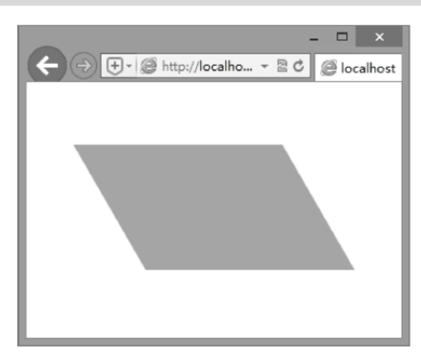


图 12.28　设计平行四边形

【示例 3】设计星形。星形的 HTML 结构也是一个<div id="star">标签，星形的实现方法比较复杂，主要是使用 transform 属性来旋转不同的边，借助:before 和:after 伪对象完成。样式代码如下，效果如图 12.29 所示。

```
<style type="text/css">
#star { /* 设计三角形后旋转，定义左顶角和右下顶角 */
    width: 0; height: 0;
    margin: 80px auto;
    color: #fc2e5a;
    position: relative;                  /* 定义定位包含框，后面生成内容根据该框定位 */
    display: block;                      /* 块显示，避免行内显示出现异常 */
    /* 设计三角形 */
    border-right: 100px solid transparent;
    border-bottom: 70px solid #fc2e5a;
    border-left: 100px solid transparent;
    /* 旋转三角形 */
    transform: rotate(35deg);
}
#star:before { /* 生成三角形，定义向上顶角 */
    content: '';                         /* 不包含内容 */
    height: 0; width: 0;
    position: absolute;                  /* 绝对定位 */
    display: block;                      /* 块显示，避免行内显示出现异常 */
    top: -45px; left: -65px;             /* 固定到顶部位置显示 */
    /* 设计三角形 */
```






```
        border-bottom: 80px solid #fc2e5a;
        border-left: 30px solid transparent;
        border-right: 30px solid transparent;
        /* 旋转三角形 */
        transform: rotate(-35deg);
}
#star:after { /* 设计三角形，然后旋转，定义右顶角和左下顶角 */
        content: '';
        width: 0; height: 0;
        position: absolute;
        display: block;
        top: 3px; left: -105px;
        border-right: 100px solid transparent;
        border-bottom: 70px solid #fc2e5a;
        border-left: 100px solid transparent;
        transform: rotate(-70deg);
}
</style>

<div id="star"></div>
```

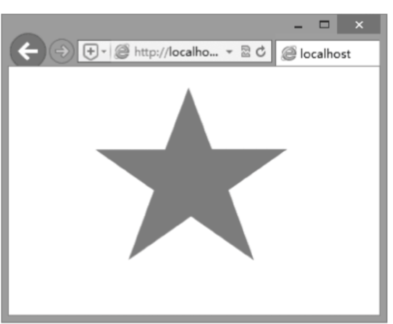


图 12.29　设计星形

【示例 4】 设计心形。心形的制作比较复杂，可以使用伪对象，分别将伪对象旋转不同的角度，并修改 transform-origin 属性来设计对象的旋转中心点。示例样式代码如下，效果如图 12.30 所示。

```
<style type="text/css">
#heart { position: relative; margin:50px auto; width:120px; }
#heart:before, #heart:after {
        content: "";
        width: 70px; height: 115px;
        position: absolute;
```





```
        background: red;
        left: 70px; top: 0;
        border-radius: 50px 50px 0 0;
        transform: rotate(-45deg);
        transform-origin: 0 100%;
    }
    #heart:after {
        left: 0;
        transform: rotate(45deg);
        transform-origin: 100% 100%;
    }
    </style>

    <div id="heart"></div>
```

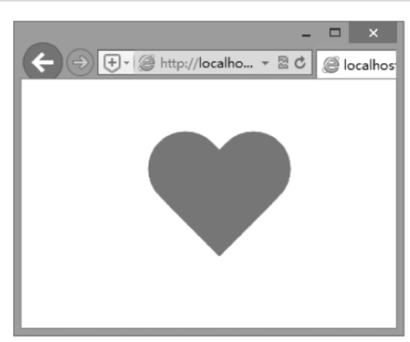


图 12.30　设计心形

提示：配合使用变形函数和伪对象选择器，用户还可以设计更多的复杂图形。



12.4.2　设计冒泡背景按钮

本例应用 CSS3 过渡动画特效，为按钮背景图像定义动态移动效果，设计当鼠标指针经过时，按钮背景绚丽多彩，不断产生冒泡的动画效果，如图 12.31 所示。



图 12.31　设计背景冒泡效果的按钮样式



【操作步骤】

第 1 步，设计按钮基本样式，代码如下。

```
.button{
    font:15px Calibri, Arial, sans-serif;
    /*半透明的文本阴影*/
    text-shadow:1px 1px 0 rgba(255,255,255,0.4);
    /*重写默认下划线的链接样式*/
    text-decoration:none !important;
    white-space:nowrap;                /*禁止文本换行显示*/
    display:inline-block;              /*行内块显示*/
    vertical-align:baseline;           /*垂直基线对齐*/
    position:relative;                 /*相对定位*/
    cursor:pointer;                    /*鼠标指针为手形*/
    padding:10px 20px;                 /*增加按钮内空间*/
    background-repeat:no-repeat;
    /*下面两个规则是回退，以防浏览器不支持多重背景*/
    background-position:bottom left;
    background-image:url('images/button_bg.png');
    /*多重背景。背景图像在颜色类中单独定义*/
    background-position:bottom left, top right, 0 0, 0 0;
    background-clip:border-box;
    /* 设计圆角 */
    border-radius:8px;
    /* 添加 1 像素的高亮效果*/
    box-shadow:0 0 1px #fff inset;
    /* 设计 CSS 过渡动画，动画属性为背景图像的位置*/
    transition:background-position 1s;
}
```

第 2 步，设计鼠标指针经过时的动态样式，代码如下。

```
.button:hover {
    background-position: top left;                /* 回退技术，兼容浏览器不支持多背景*/
    background-position: top left, bottom right, 0 0, 0 0;
}
```

第 3 步，设计激活时按钮下沉的样式，代码如下。

```
.button:active { bottom: -1px; }
```



第 4 步，设计按钮大小号类样式，代码如下。



Note

```
.button.button_big { font-size: 30px; }
.button.button_medium { font-size: 18px; }
.button.button_small { font-size: 13px; }
```

第 5 步，设计圆角按钮类样式，代码如下。

```
.button.button_rounded { border-radius: 4em;}
```

第 6 步，设计按钮主题类样式，代码如下。

```
.button_blue.button {
    color: #0f4b6d !important;
    border: 1px solid #84acc3 !important;
    /*  回退背景色颜色  */
    background-color: #48b5f2;
    /*  定义多背景图  */
    background-image: url('images/button_bg.png'), url('images/button_bg.png'), radial-gradient( center bottom, circle, rgba(89,208,244,1) 0, rgba(89,208,244,0) 100px), linear-gradient(#4fbbf7, #3faeeb);
}
.button_blue.button:hover { /*  定义鼠标指针经过时多背景图的样式  */
    background-color: #63c7fe;
    background-image: url('images/button_bg.png'), url('images/button_bg.png'), radial-gradient( center bottom, circle, rgba(109,217,250,1) 0, rgba(109,217,250,0) 100px), linear-gradient(#63c7fe, #58bef7);
}
```

12.4.3　设计动画效果菜单



视频讲解

本例利用 CSS3 过渡动画设计一个界面切换的导航菜单，当鼠标指针经过菜单项时，会以动画形式从中文界面缓慢翻转到英文界面，或者从英文界面翻转到中文界面，效果如图 12.32 所示。



图 12.32　设计动画翻转菜单样式

【操作步骤】

第 1 步，设计菜单结构。在每个菜单项（<div class="menu1">）中包含两个子标签：<div class="one">和<div class="two">，设计菜单项仅显示一个子标签，当鼠标指针经过时，翻转显示另一个子





标签，代码如下。

```
<div>
    <div class="menu1">
        <div class="one"><a href="#">首页</a></div>
        <div class="two"><a href="#">Home</a></div>
    </div>
    <div class="menu1">
        <div class="one"><a href="#">新闻</a></div>
        <div class="two"><a href="#">News</a></div>
    </div>
    <div class="menu1">
        <div class="one"><a href="#">关于</a></div>
        <div class="two"><a href="#">About</a></div>
    </div>
</div>
```

第 2 步，设计菜单项样式：固定大小、相对定位，禁止内容溢出，向左浮动，定义并列显示，代码如下。

```
.menu1 {
    width: 100px; height: 30px;
    position: relative;
    font-family: 微软雅黑; font-size: 12px; color: #fff;
    overflow: hidden;
    float: left;
}
```

第 3 步，设计每个菜单项中子标签<div class="one">和 <div class="two">的样式。定义它们与菜单项相同大小，这样就只能够显示一个子标签；为了方便控制，定义它们为绝对定位，包含文本水平居中和垂直居中，最后定义过渡动画时间为 0.3s，加速到减速显示，代码如下。

```
.menu1 div {
    width: 100px; height: 30px;
    line-height: 30px;    text-align: center;
    position: absolute;
    transition: all 0.3s ease-in-out;
}
```

第 4 步，设计过渡动画样式。本例设计过渡演示属性为 left、top 和 bottom，当鼠标指针经过时，改变定位属性的值，实现菜单项动态翻转效果，代码如下。





```
.menu1 .one {
    top: 0; left: 0;
    z-index: 1;
    background: #63C; color: #FFF;
}
.menu1:hover .one { top: -30px; left: 0;}
.menu1 .two {
    bottom: -30px; left: 0;
    z-index: 2;
    background: #f50; color: #FFF;
}
.menu1:hover .two { bottom: 0px; left: 0;}
```



视频讲解

12.4.4 设计照片特效

本例使用 CSS3 阴影、透明效果，以及变换，让图片随意显示，当鼠标指针移动到图片上时，会自动放大并垂直摆放，演示效果如图 12.33 所示。



图 12.33 设计挂图效果

主要代码如下。

```
<style type="text/css">
ul.polaroids li { display: inline;}
ul.polaroids a {
    display: inline; float: left;
    margin: 0 0 50px 60px; padding: 12px;
    text-align: center;
    text-decoration: none; color: #333;
     /*为图片外框设计阴影效果 */
    box-shadow: 0 3px 6px rgba(0, 0, 0, .25);
     /*设置过渡动画：过渡属性为 transform，时长为 0.15s，线性渐变 */
    transition: -webkit-transform .15s linear;
    /*顺时针旋转 2° */
```





```
        transform: rotate(-2deg);
}
ul.polaroids img { /*统一图片基本样式 */
        display: block;
        height: 100px;
        border: none;
        margin-bottom: 12px;
}
        /*利用图片的 tittle 属性，添加图片显示标题 */
ul.polaroids a:after { content: attr(title);}
        /*为偶数图片倾斜显示*/
ul.polaroids li:nth-child(even) a {
        transform: rotate(10deg);                /*逆时针旋转 10° */
}
ul.polaroids li a:hover {
          /*放大对象 1.25 倍 */
        transform: scale(1.25);
        box-shadow: 0 3px 6px rgba(0, 0, 0, .5);
}
</style>

<ul class="polaroids">
        <li> <a href="1" title="相识"> <img src="images/1.jpg" alt="相识"> </a> </li>
        <li> <a href="2" title="相知"> <img src="images/2.jpg" alt="相知"> </a> </li>
        <li> <a href="3" title="自信"> <img src="images/3.jpg" alt="自信"> </a> </li>
</ul>
```

12.4.5　设计立体盒子



【示例 1】本示例使用 2D 多重变换制作一个正方体。代码如下，演示效果如图 12.34 所示。

```
<style type="text/css">
body{padding:20px 0 0 100px;}
.side {
        height: 100px; width: 100px;
        position: absolute;
        font-size: 20px; font-weight: bold; line-height: 100px; text-align: center; color: #fff;
        text-shadow: 0 -1px 0 rgba(0,0,0,0.2);
        text-transform: uppercase;
}
```





Note

```
.top {/*顶面*/
    background: red;
    transform: rotate(-45deg) skew(15deg, 15deg);
}
.left {/*左侧面*/
    background: blue;
    transform: rotate(15deg) skew(15deg, 15deg) translate(-50%, 100%);
}
.right {/*右侧面*/
    background: green;
    transform: rotate(-15deg) skew(-15deg, -15deg) translate(50%, 100%);
}
</style>

<div class="side top">顶面</div>
<div class="side left">左侧面</div>
<div class="side right">右侧面</div>
```

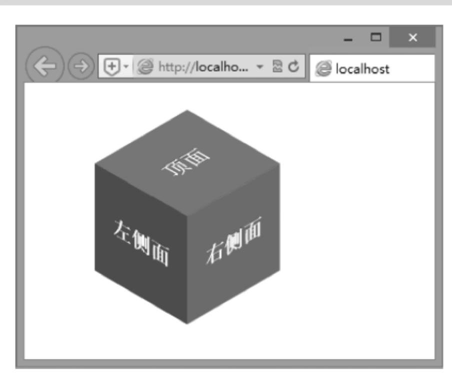


图 12.34　设计 2D 变换盒子

【示例 2】本示例使用 3D 多重变换制作一个正方体。代码如下，演示效果如图 12.35 所示。

```
<style type="text/css">
.stage {/*定义画布样式 */
    width: 300px; height: 300px; margin: 100px auto; position: relative;
    perspective: 300px;
}
/*定义盒子包含框样式 */
.container { transform-style: preserve-3d;}
/*定义盒子六面基本样式 */
.side {
```





```
        background: rgba(255,0,0,0.3);
        border: 1px solid red;
        font-size: 60px; font-weight: bold; color: #fff; text-align: center;
        height: 196px; line-height: 196px; width: 196px;
        position: absolute;
        text-shadow: 0 -1px 0 rgba(0,0,0,0.2);
        text-transform: uppercase;
}
.front {/*使用 3D 变换制作前面 */
        transform: translateZ(100px);
}
.back {/*使用 3D 变换制作后面 */
        transform: rotateX(180deg) translateZ(100px);
}
.left {/*使用 3D 变换制作左面 */
        transform: rotateY(-90deg) translateZ(100px);
}
.right {/*使用 3D 变换制作右面 */
        transform: rotateY(90deg) translateZ(100px);
}
.top {/*使用 3D 变换制作顶面 */
        transform: rotateX(90deg) translateZ(100px);
}
.bottom {/*使用 3D 变换制作底面 */
        transform: rotateX(-90deg) translateZ(100px);
}
</style>

<div class="stage">
        <div class="container">
                <div class="side front">前面</div>
                <div class="side back">背面</div>
                <div class="side left">左面</div>
                <div class="side right">右面</div>
                <div class="side top">顶面</div>
                <div class="side bottom">底面</div>
        </div>
</div>
```





Note

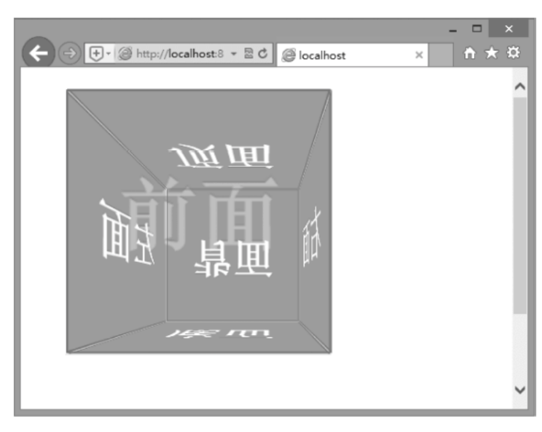


图 12.35　设计 3D 盒子



视 频 讲 解

12.4.6　旋转盒子

本节示例以 12.4.5 节示例为基础，使用 animation 属性设计盒子旋转显示。

【示例 1】本例使用 2D 制作一个正方体，然后设计它在鼠标指针经过时沿 y 轴旋转，演示效果如图 12.36 所示。

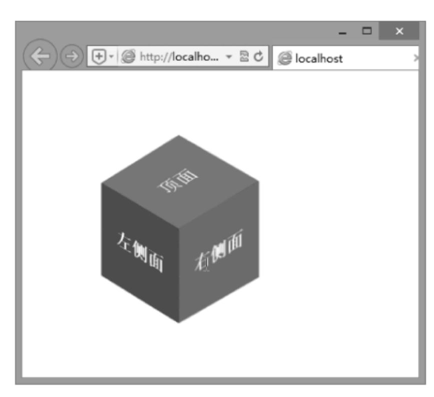


图 12.36　设计旋转的 3D 盒子

第 1 步，复制上节示例 index1.html。在 HTML 结构中为盒子添加两层包含框，代码如下。

```
<div class="stage s1">
    <div class="container">
        <div class="side top">Top</div>
        <div class="side left">Left</div>
        <div class="side right">Right</div>
    </div>
</div>
```

第 2 步，在内部样式表中定义关键帧，代码如下。





```
/*定义关键帧动画 */
@keyframes spin{/*标准模式 */
    0%{transform:rotateY(0deg)}
    100%{transform:rotateY(360deg)}
}
```

第 3 步，设计 3D 变换的透视距离和变换类型，即启动 3D 变换，代码如下。

```
/*定义盒子所在画布框的样式 */
.stage { perspective: 1200px;}
/*定义盒子包含框样式 */
.container { transform-style: preserve-3d;}
```

第 4 步，定义动画触发方式，代码如下。

```
/*定义鼠标指针经过盒子时，触发线性变形动画，动画时长 5s，持续播放 */
.container:hover{ animation:spin 5s linear infinite;}
```

本例完整代码请参考本书源代码。

【示例 2】本例使用 3D 制作一个正方体，然后设计它在鼠标指针经过时沿 y 轴旋转，演示效果如图 12.37 所示。



图 12.37　设计旋转的 3D 盒子

第 1 步，在内部样式表中定义关键帧，代码如下。

```
/*定义关键帧动画 */
@keyframes spin {
    0% {transform:rotateY(0deg)}
    100% {transform:rotateY(360deg)}
}
```

第 2 步，设计 3D 变换的透视距离和变换类型，即启动 3D 变换，代码如下。





Note

```
/*定义画布样式 */
.stage { perspective: 300px; }
/*定义盒子包含框样式 */
.container { transform-style: preserve-3d; }
```

第 3 步，定义动画触发方式，代码如下。

```
/*定义鼠标指针经过时触发盒子旋转动画 */
.container:hover { animation: spin 5s linear infinite;}
```

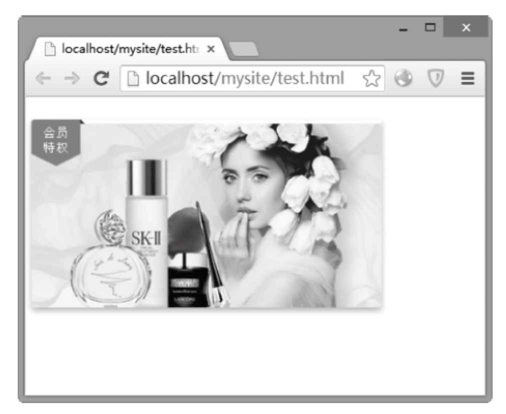
本例完整代码请参考本书源代码。



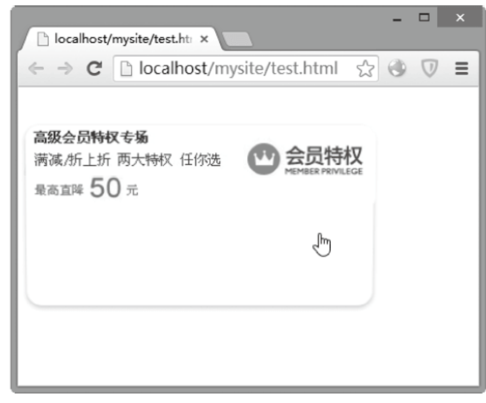
视频讲解

12.4.7 设计翻转广告

本例设计当鼠标指针移动到产品图片上时，产品信息翻转滑出的效果，如图 12.38 所示。在默认状态下只显示产品图片，而产品信息隐藏不可见。当用户将鼠标指针移动到产品图像上时，产品图像慢慢往上旋转使产品信息展示出来，而产品图像慢慢隐藏起来，看起来就像是一个旋转的盒子。



默认状态



翻转状态

图 12.38 设计 3D 翻转广告牌

主要代码如下。

```
<style type="text/css">
/*定义包含框样式 */
.wrapper {
    display: inline-block; width: 345px; height: 186px; margin: 1em auto; cursor: pointer; position: relative;
/*定义 3D 元素距视图的距离 */
    perspective: 4000px;
}
/*定义旋转元素样式：3D 动画，动画时长 0.6 秒 */
.item {
```





```
        height: 186px;
        transform-style: preserve-3d;
        transition: transform .6s;
}
*定义鼠标指针经过时触发动画，并定义旋转形式 */
.item:hover {
        transform: translateZ(-50px) rotateX(95deg);
}
.item:hover img {box-shadow: none; border-radius: 15px;}
.item:hover .information { box-shadow: 0px 3px 8px rgba(0,0,0,0.3); border-radius: 15px;}
/*定义广告图的动画形式和样式 */
.item>img {
        display: block; position: absolute; top: 0; border-radius: 3px;box-shadow: 0px 3px 8px rgba(0,0,0,0.3);
        transform: translateZ(50px);
        transition: all .6s;
}
/*定义广告文字的动画形式和样式 */
.item .information {
        position: absolute; top: 0; height: 186px; width: 345px; border-radius: 15px;
        transform: rotateX(-90deg) translateZ(50px);
        transition: all .6s;
}
</style>

<div class="wrapper">
        <div class="item">
                <img src="images/1.png" />
                <span class="information"><img src="images/2.png" /></span>
        </div>
</div>
```

12.4.8　设计跑步效果



本例设计一个跑步动画效果，主要使用 CSS3 帧动画控制一张序列人物跑步的背景图像，在页面固定“镜头”中快速切换实现动画效果，如图 12.39 所示。

【操作步骤】

第 1 步，设计舞台场景结构。新建 HTML 文档，保存为 index1.html。输入以下代码。

```
<div class="charector-wrap " id="js_wrap">
        <div class="charector"></div>
</div>
```








图 12.39 设计跑步的小人

第 2 步，设计舞台基本样式，代码如下。其中导入的小人图片是一个序列跑步人物，如图 12.40 所示。

```
.charector-wrap {
    position: relative;
    width: 180px;
    height: 300px;
    left: 50%;
    margin-left: -90px;
}
.charector{
    position: absolute;
    width: 180px;
    height:300px;
    background: url(img/charector.png) 0 0 no-repeat;
}
```









图 12.40 小人序列集合

本例主要设计任务就是让序列小人仅显示一个，然后通过 CSS3 动画，让他们快速闪现在指定的限定框中。

第 3 步，设计动画关键帧，代码如下。

```
@keyframes person-normal{/*跑步动画名称 */
    0% {background-position: 0 0;}
```





```
        14.3% {background-position: -180px 0;}
        28.6% {background-position: -360px 0;}
        42.9% {background-position: -540px 0;}
        57.2% {background-position: -720px 0;}
        71.5% {background-position: -900px 0;}
        85.8% {background-position: -1080px 0;}
        100% {background-position: 0 0;}
}
```

第 4 步，设置动画属性，代码如下。

```
.charector{
        animation-iteration-count: infinite;         /* 动画无限播放 */
        animation-timing-function:step-start;        /* 马上跳到动画每一结束帧的状态 */
}
```

第 5 步，启动动画并设置动画频率，代码如下。

```
/* 启动动画并控制跑步动作频率*/
.charector{
        animation-name: person-normal;
        animation-duration: 800ms;
}
```

12.4.9　设计折叠面板



本例使用 CSS3 的目标伪类（:target）设计折叠面板效果，没有使用 JavaScript 脚本，使用过渡属性设计滑动效果，折叠动画效果如图 12.41 所示。



图 12.41　设计折叠面板





主要代码如下。

```
<style type="text/css">
/* 定义折叠框外框样式 */
.accordion {
    background: #eee;
    border: 1px solid #999;
    margin: 2em;}
/* 定义折叠框标题栏样式 */
.accordion h2 {
    margin: 0;
    padding: 12px 0;
    background:#CCC}
/* 定义折叠框内容框样式 */
.accordion .section {
    border-bottom: 1px solid #ccc;
    background: #fff;}
/* 定义折叠框选项标题栏样式*/
.accordion h3 {
    margin:0;
    padding:0;
    background: #eee;
    padding:3px 1em;}
/* 定义折叠框选项标题栏超链接样式*/
.accordion h3 a {
    font-weight: normal;
    text-decoration:none;}
/* 当获得目标焦点时，粗体显示选项标题栏文字*/
.accordion :target h3 a { font-weight: bold; }
/* 选项栏标题对应的选项子框样式 */
.accordion h3 + div {
    height: 0;
    padding:0 1em;
    overflow: hidden;
/*定义过渡对象为高度，过渡时间为 0.3 秒，渐显显示*/
    transition: height 0.3s ease-in;}
.accordion h3 + div img { margin:4px; }
/* 当获得目标焦点时的子选项内容框样式 */
```



```
.accordion :target h3 + div {
/*当获取目标之后，高度为 300 像素*/
    height:300px;
    overflow:auto;}
</style>

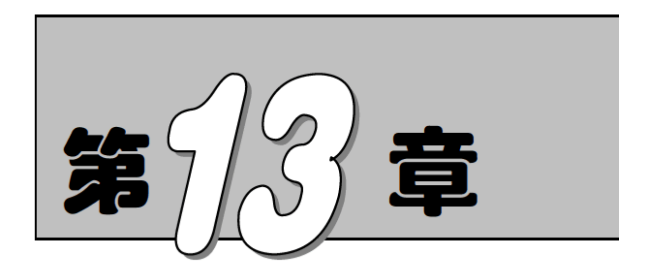
<div class="accordion">
    <h2>我爱买</h2>
    <div id="one" class="section">
        <h3> <a href="#one">爱逛</a> </h3>
        <div><img src="images/11.png"></div>
    </div>
    <div id="two" class="section">
        <h3> <a href="#two">爱美丽</a> </h3>
        <div><img src="images/22.png"></div>
    </div>
    <div id="three" class="section">
        <h3> <a href="#three">爱吃</a> </h3>
        <div><img src="images/33.png"></div>
    </div>
</div>
```

12.5　在线练习

练习 CSS3 动画一般设计方法，培养灵活应用交互式动态样式的基本能力。感兴趣的读者可以扫码练习。



在线练习



使用 CSS3 设计响应式页面

2017 年 9 月，W3C 发布了媒体查询（Media Query Level 4）候选推荐标准规范，它扩展了已经发布的媒体查询的功能。该规范用于 CSS 的@media 规则，可以为文档设定特定条件的样式，也可用于 HTML、JavaScript 等语言中。

权威参考：http://www.w3.org/TR/css3-mediaqueries/。

【学习要点】

- 了解 CSS3 媒体类型。
- 正常使用媒体查询的条件规则。
- 设计响应不同设备的网页布局。





13.1　媒体查询基础

媒体查询可以根据设备特性，如屏幕宽度、高度、设备方向（横向或纵向），为设备定义独立的 CSS 样式表。一个媒体查询由一个可选的媒体类型和零个或多个限制范围的表达式组成，如宽度、高度和颜色。

13.1.1　媒体类型和媒体查询

CSS 2 提出媒体类型（Media Type）的概念，它允许为样式表设置限制范围的媒体类型。例如，仅供打印的样式表文件、仅供手机渲染的样式表文件、仅供电视渲染的样式表文件等，具体说明如表 13.1 所示。

表 13.1　CSS 媒体类型

类　型	支持的浏览器	说　明
aural	Opera	用于语音和音乐合成器
braille	Opera	用于触觉反馈设备
handheld	Chrome,Safari,Opera	用于小型或手持设备
print	所有浏览器	用于打印机
projection	Opera	用于投影图像，如幻灯片
screen	所有浏览器	用于屏幕显示器
tty	Opera	用于使用固定间距字符格的设备，如电传打字机和终端
tv	Opera	用于电视类设备
embossed	Opera	用于凸点字符（盲文）印刷设备
speech	Opera	用于语音类型
all	所有浏览器	用于所有媒体设备类型

通过<HTML>标签 media 属性定义样式表的媒体类型，具体方法如下。

☑　定义外部样式表文件的媒体类型

```
<link href="csss.css" rel="stylesheet" type="text/css" media="handheld" />
```

☑　定义内部样式表文件的媒体类型

```
<style type="text/css" media="screen">
...
</style>
```

CSS3 在媒体类型基础上，提出了 Media Queries（媒体查询）的概念。媒体查询比 CSS2 的媒体类型功能更强大、更完善。两者主要区别是：媒体查询是一个值或一个范围的值，而媒体类型仅仅




Note

是设备的匹配。媒体类型可以帮助用户获取以下数据。

☑ 浏览器窗口的宽和高
☑ 设备的宽和高
☑ 设备的手持方向（横向还是竖向）
☑ 分辨率

例如下面这条导入外部样式表的语句：

```
<link rel="stylesheet" media="screen and (max-width: 600px)" href="small.css" />
```

在 media 属性中设置媒体查询的条件(max-width: 600px)：当屏幕宽度小于或等于 600px 时，则调用 small.css 样式表来渲染页面。

13.1.2 使用@media

CSS3 使用@media 规则定义媒体查询，简化语法格式如下。

```
@media [only | not]? <media_type> [and <expression>]* | <expression> [and <expression>]*{
    /* CSS 样式列表 */
}
```

参数简单说明如下。

☑ <media_type>：指定媒体类型，具体说明参考表 13.1。
☑ <expression>：指定媒体特性，放在一对圆括号中，如(min-width:400px)。
☑ 逻辑运算符，如 and（逻辑与）、not（逻辑否）、only（兼容设备）等。

媒体特性包括 13 种，接受单个的逻辑表达式作为值或者没有值。大部分特性接受 min 或 max 的前缀，用来表示大于等于或者小于等于的逻辑，以此避免使用大于号（>）和小于号（<）字符。

下面是各种媒体特性的简单说明。

1. *颜色（color）*

指定输出设备每个像素单元的比特值，接受 min/max 前缀。例如，每个颜色单元至少有 4 个比特的设备应用样式表，代码如下。

```
@media all and (min-color: 4) { ... }
```

2. *颜色索引（color-index）*

指定输出设备颜色查询表中的条目数量，接受 min/max 前缀。例如，向所有使用索引颜色的设备应用样式表，代码如下。

```
@media all and (color-index) { ... }
```

3. *宽高比（aspect-ratio）*

指定输出设备目标显示区域的宽高比。取值包含两个以“/”分隔的正整数，代表水平像素数与垂直像素数的比例，接受 min/max 前缀。例如，下面为显示区域宽高比至少为 1∶1 的设备选择了一





个特殊的样式表，代码如下。

```
@media screen and (min-aspect-ratio: 1/1) { ... }
```

4. 设备宽高比（device-aspect-ratio）

指定输出设备的宽高比，接受 min/max 前缀。例如，下面为宽屏设备选择了一个特殊的样式表，代码如下。

```
@media screen and (device-aspect-ratio: 16/9), screen and (device-aspect-ratio: 16/10) { ... }
```

5. 设备高度（device-height）

指定输出设备的高度，接受 min/max 前缀。例如，在最大宽度 800px 的屏幕上应用样式表，代码如下。

```
<link rel="stylesheet" media="screen and (max-device-width: 799px)" />
```

6. 设备宽度（device-width）

指定输出设备的宽度，接受 min/max 前缀。

7. 网格（grid）

判断输出设备是网格设备还是位图设备。如果设备是基于网格的（如电传打字机），该值为 1，否则为 0。例如，向一个 15 字符宽度或更窄的手持设备应用样式，代码如下。

```
@media handheld and (grid) and (max-width: 15em) { ... }
```

8. 高度（height）

指定输出设备渲染区域的高度，接受 min/max 前缀。

9. 黑白（monochrome）

指定一个黑白设备每个像素的比特数，接受 min/max 前缀。例如，向所有黑白设备应用样式表，代码如下。

```
@media all and (monochrome) { ... }
```

10. 方向（orientation）

指定设备处于横屏（宽度大于高度）模式还是竖屏（高度大于宽度）模式。取值包括 landscape（横屏）和 portrait（竖屏）。例如，向竖屏设备应用样式表，代码如下。

```
@media all and (orientation: portrait) { ... }
```

11. 分辨率（resolution）

指定输出设备的分辨率，接受 min/max 前缀。例如，为每英寸至多 300 点的打印机应用样式，代码如下。




Note

```
@media print and (min-resolution: 300dpi) { ... }
```

12. 扫描（scan）

指定电视输出设备的扫描过程，取值包括 progressive 和 interlace（交错）。例如，向以顺序方式扫描的电视机上应用样式表，代码如下。

```
@media tv and (scan: progressive) { ... }
```

13. 宽度（width）

指定输出设备渲染区域的宽度，接受 min/max 前缀。例如，向宽度在 500～800px 的屏幕应用样式表，代码如下。

```
@media screen and (min-width: 500px) and (max-width: 800px) { ... }
```

在 CSS 样式的开头必须定义@media 关键字，然后指定媒体类型，再指定媒体特性。媒体特性的格式与样式的格式相似，分为两部分，由冒号分隔，冒号前指定媒体特性，冒号后指定该特性的值。

例如，下面语句指定了当设备显示屏幕宽度小于 640px 时所使用的样式。

```
@media screen and (max-width: 639px) {
    /*样式代码*/
}
```

可以使用多个媒体查询将同一个样式应用于不同的媒体类型和媒体特性中，媒体查询之间用逗号分隔，类似于选择器分组，代码如下。

```
@media handheld and (min-width:360px),screen and (min-width:480px) {
    /*样式代码*/
}
```

可以在表达式中加上 not、only 和 and 等逻辑运算符，代码如下。

```
//下面样式代码将被使用在除便携设备之外的其他设备或非彩色便携设备中
@media not handheld and (color) {
    /*样式代码*/
}
//下面样式代码将被使用在所有非彩色设备中
@media all and (not color) {
    /*样式代码*/
}
```

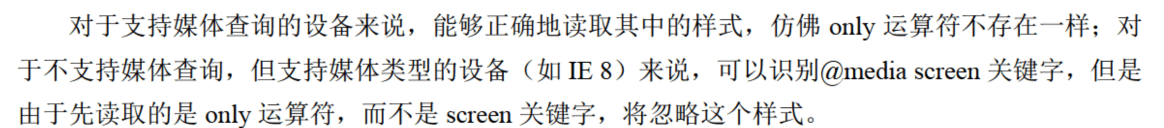
only 运算符能够让那些不支持媒体查询但是支持媒体类型的设备忽略表达式中的样式，代码如下。

```
@media only screen and (color) {
```

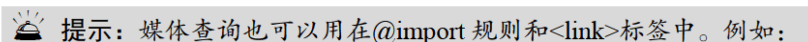


```
    /*样式代码*/
}
```





对于支持媒体查询的设备来说，能够正确地读取其中的样式，仿佛 only 运算符不存在一样；对于不支持媒体查询，但支持媒体类型的设备（如 IE 8）来说，可以识别@media screen 关键字，但是由于先读取的是 only 运算符，而不是 screen 关键字，将忽略这个样式。



提示：媒体查询也可以用在@import 规则和<link>标签中。例如：

```
@import url(example.css) screen and (width:800px);
//下面代码定义了如果页面通过屏幕呈现，且屏幕宽度不超过 480px，则加载 shetland.css 样式表
<link rel="stylesheet" type="text/css" media="screen and (max-device-width: 480px)" href="shetland.css" />
```

13.1.3　应用@media

【示例 1】and 运算符用于符号两边规则均满足条件的匹配，代码如下。

```
@media screen and (max-width : 600px) {
    /*匹配宽度小于等于 600px 的屏幕设备*/
}
```

【示例 2】not 运算符用于取非，所有不满足该规则的均匹配，代码如下。

```
@media not print {
    /*匹配除了打印机以外的所有设备*/
}
```

注意：not 仅应用于整个媒体查询，代码如下。

```
@media not all and (max-width : 500px) {}
/*等价于*/
@media not (all and (max-width : 500px)) {}
/*而不是*/
@media (not all) and (max-width : 500px) {}
```

在逗号媒体查询列表中，not 仅会否定它所在的媒体查询，而不影响其他的媒体查询。

如果在复杂的条件中使用 not 运算符，要显式添加小括号，避免歧义。

【示例 3】逗号（,）相当于 or 运算符，用于两边有一条满足则匹配，代码如下。

```
@media screen , (min-width : 800px) {
    /*匹配屏幕或者宽度大于等于 800px 的设备*/
}
```





Note

【示例 4】在媒体类型中，all 是默认值，匹配所有设备，代码如下。

```
@media all {
    /*可以过滤不支持 media 的浏览器*/
}
```

常用的媒体类型还有 screen 匹配屏幕显示器、print 匹配打印输出等，更多媒体类型可以参考表 13.1。

【示例 5】使用媒体查询时，必须要加括号，一个括号就是一个查询，代码如下。

```
@media (max-width : 600px) {
    /*匹配界面宽度小于等于 600px 的设备*/
}
@media (min-width : 400px) {
    /*匹配界面宽度大于等于 400px 的设备*/
}
@media (max-device-width : 800px) {
    /*匹配设备（不是界面）宽度小于等于 800px 的设备*/
}
@media (min-device-width : 600px) {
    /*匹配设备（不是界面）宽度大于等于 600px 的设备*/
}
```

提示： 在设计手机网页时，应该使用 device-width/device-height，因为手机浏览器默认会对页面进行一些缩放，如果按照设备宽高来进行匹配，会更接近预期的效果。

【示例 6】媒体查询允许相互嵌套，这样可以优化代码，避免冗余，代码如下。

```
@media not print {
    /*通用样式*/
    @media (max-width:600px) {
    /*此条匹配宽度小于等于 600px 的非打印机设备 */
    }
    @media (min-width:600px) {
    /*此条匹配宽度大于等于 600px 的非打印机设备 */
    }
}
```

【示例 7】在设计响应式页面时，用户应该根据实际需要，先确定自适应分辨率的阈值，也就是页面响应的临界点，代码如下。





```
@media (min-width: 768px){
    /* 大于等于 768px 的设备 */
}
@media (min-width: 992px){
    /* 大于等于 992px 的设备 */
}
@media (min-width: 1200){
    /* 大于等于 1200px 的设备 */
}
```

注意： 下面样式顺序是错误的，因为后面的查询范围将覆盖前面的查询范围，导致前面的媒体查询失效。

```
@media (min-width: 1200){ }
@media (min-width: 992px){ }
@media (min-width: 768px){  }
```

因此，当我们使用 min-width 媒体特性时，应该按从小到大的顺序设计各个阈值。同理，如果使用 max-width，就应该按从大到小的顺序设计各个阈值，代码如下。

```
@media (max-width: 1199){
    /* <=1199px 的设备 */
}
@media (max-width: 991px){
    /* <=991px 的设备 */
}
@media (max-width: 767px){
    /* <=767px 的设备 */
}
```

【示例 8】 用户可以创建多个样式表，以适应不同媒体类型的宽度范围。当然，更有效率的方法是将多个媒体查询整合在一个样式表文件中，这样可以减少请求的数量，代码如下。

```
@media only screen  and (min-device-width : 320px)  and (max-device-width : 480px) {
    /*样式列表 */
}
@media only screen  and (min-width : 321px) {
    /*样式列表 */
}
@media only screen  and (max-width : 320px) {
```






```
        /*样式列表 */
    }
```

【示例 9】如果从资源的组织和维护的角度考虑，可以选择使用多个样式表的方式来实现媒体查询，这样做更高效，代码如下。

```
<link rel="stylesheet" media="screen and (max-width: 600px)" href="small.css" />
<link rel="stylesheet" media="screen and (min-width: 600px)" href="large.css" />
<link rel="stylesheet" media="print" href="print.css" />
```

【示例 10】使用 orientation 属性可以判断设备屏幕当前是横屏（值为 landscape）还是竖屏（值为 portrait），代码如下。

```
@media screen and (orientation: landscape) {
    .iPadLandscape {
        width: 30%;
        float: right;
    }
}
@media screen and (orientation: portrait) {
    .iPadPortrait {clear: both;}
}
```

不过 orientation 属性只在 iPad 上有效，对于其他可转屏的设备（如 iPhone），可以使用 min-device-width 和 max-device-width 来变通实现。

【扩展】

媒体查询仅是一种纯 CSS 方式实现响应式 Web 设计的方法，用户还可以使用 JavaScript 库来实现同样的设计。例如，下载 css3-mediaqueries.js（http://code.google.com/p/css3-mediaqueries-js/），然后在页面中调用。对于老式浏览器（如 IE 6、IE 7、IE 8）可以考虑使用 css3-mediaqueries.js 进行兼容，代码如下。

```
<!–[if lt IE 9]>
<script src=”http://css3-mediaqueries-js.googlecode.com/svn/trunk/css3-mediaqueries.js”></script>
<![endif]–>
```

【示例 11】下面代码演示了如何使用 jQuery 来检测浏览器宽度，并为不同的视口调用不同的样式表，代码如下。

```
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
    $(window).bind("resize", resizeWindow);
```





```
        function resizeWindow(e){
            var newWindowWidth = $(window).width();
            if(newWindowWidth < 600){
                $("link[rel=stylesheet]").attr({href : "mobile.css"});
            }
            else if(newWindowWidth > 600){
                $("link[rel=stylesheet]").attr({href : "style.css"});
            }
        }
    });
</script>
```

13.2　案例实战

本节将通过几个案例练习 CSS3 媒体查询的网页应用。

13.2.1　判断显示屏幕宽度



视频讲

本例将演示如何正确使用@media 规则判断当前视口宽度的范围。示例代码如下。

```
<style type="text/css">
.wrapper {/* 定义测试条的样式 */
    padding: 5px 10px; margin: 40px;
    text-align:center; color:#999;
    border: solid 1px #999;
}
.viewing-area span {/* 默认情况下隐藏提示文本信息 */
    color: #666;
    display: none;
}
/* 应用于移动设备，且设备最大宽度为 480 像素 */
@media screen and (max-device-width: 480px) {
    .a { background: #ccc;}
}
/* 显示屏幕宽度小于等于 600 像素 */
@media screen and (max-width: 600px) {
    .b {
        background: red; color:#fff;
```



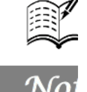

Note

```
            border: solid 1px #000;
        }
        span.lt600 { display: inline-block; }
}
/*  显示屏幕宽度介于 600 像素到 900 像素之间*/
@media screen and (min-width: 600px) and (max-width: 900px) {
        .c {
            background: red; color:#fff;
            border: solid 1px #000;
        }
        span.bt600-900 { display: inline-block; }
}
/*  显示屏幕宽度大于等于 900 像素*/
@media screen and (min-width: 900px) {
        .d {
            background: red;    color:#fff;
            border: solid 1px #000;
        }
        span.gt900 { display: inline-block; }
}
</style>

<div class="wrapper a">设备最大宽度为 480 像素。</div>
<div class="wrapper b">显示屏幕宽度小于等于 600 像素</div>
<div class="wrapper c">显示屏幕宽度介于 600 像素到 900 像素之间</div>
<div class="wrapper d">显示屏幕宽度大于等于 900 像素</div>
<p class="viewing-area">
        <strong>当前显示屏幕宽度：</strong>
        <span class="lt600">小于等于 600px</span>
        <span class="bt600-900">介于 600px - 900px 之间</span>
        <span class="gt900">大于等于 900px</span>
</p>
```

示例设计当显示屏幕宽度小于等于 600 像素时，高亮显示<div class="wrapper b">测试条，并在底部显示提示信息：小于等于 600px；当显示屏幕宽度介于 600 像素和 900 像素之间时，高亮显示<div class="wrapper c">测试条，并在底部显示提示信息：介于 600px-900px 之间；显示屏幕宽度大于等于 900 像素时，高亮显示<div class="wrapper d">测试条，并在底部显示提示信息：大于等于 900px；当设备宽度小于等于 480 像素时，则高亮显示<div class="wrapper a">测试条。演示效果如图 13.1 所示。







显示屏幕宽度小于等于600像素



显示屏幕宽度介于600像素到900像素之间



显示屏幕宽度大于等于900像素

图 13.1　使用@media 规则

13.2.2　设计响应式版式



本例在页面中设计如下 3 个栏目。

☑　<div id="main">：主要内容栏目。

☑　<div id="sub">：次要内容栏目。

☑　<div id="sidebar">：侧边栏栏目。

构建的页面结构如下。

```
<div id="container">
    <div id="wrapper">
        <div id="main">
            <h1>水调歌头・明月几时有</h1>
            <h2>苏轼</h2>
            <p>……</p>
        </div>
        <div id="sub">
            <h2>宋词精选</h2>
            <ul>
                <li>……</li>
            </ul>
```






```
            </div>
        </div>
        <div id="sidebar">
            <h2>词人列表</h2>
            <ul>
                <li>……</li>
            </ul>
        </div>
    </div>
```

设计页面能够自适应屏幕宽度，呈现不同的版式布局。当显示屏幕宽度在 999 像素以上时，让 3 个栏目并列显示；当显示屏幕宽度在 639 像素以上、1000 像素以下时，设计两栏目显示；当显示屏幕宽度在 640 像素以下时，让 3 个栏目堆叠显示。代码如下。

```
<style type="text/css">
/*  默认样式  */
/*  网页宽度固定，并居中显示  */
#container { width: 960px; margin: auto;}
/*主体宽度  */
#wrapper {width: 740px; float: left;}
/*设计 3 栏并列显示*/
#main {width: 520px; float: right;}
#sub { width: 200px; float: left;}
#sidebar { width: 200px; float: right;}
/*  窗口宽度在 999 像素以上  */
@media screen and (min-width: 1000px) {
    /* 3 栏显示*/
    #container { width: 1000px; }
    #wrapper { width: 780px; float: left; }
    #main {width: 560px; float: right; }
    #sub { width: 200px; float: left; }
    #sidebar { width: 200px; float: right; }
}
/*  窗口宽度在 639 像素以上、1000 像素以下  */
@media screen and (min-width: 640px) and (max-width: 999px) {
    /* 2 栏显示  */
    #container { width: 640px; }
    #wrapper { width: 640px; float: none; }
    .height { line-height: 300px; }
```





```
        #main { width: 420px; float: right; }
        #sub {width: 200px; float: left; }
        #sidebar {width: 100%; float: none; }
}
/*  窗口宽度在 640 像素以下  */
@media screen and (max-width: 639px) {
        /*   1 栏显示    */
        #container { width: 100%; }
        #wrapper { width: 100%; float: none; }
        #main {width: 100%; float: none; }
        #sub { width: 100%; float: none; }
        #sidebar { width: 100%; float: none; }
}
</style>
```

当显示屏幕宽度在 999 像素以上时，3 栏并列显示，预览效果如图 13.2 所示。

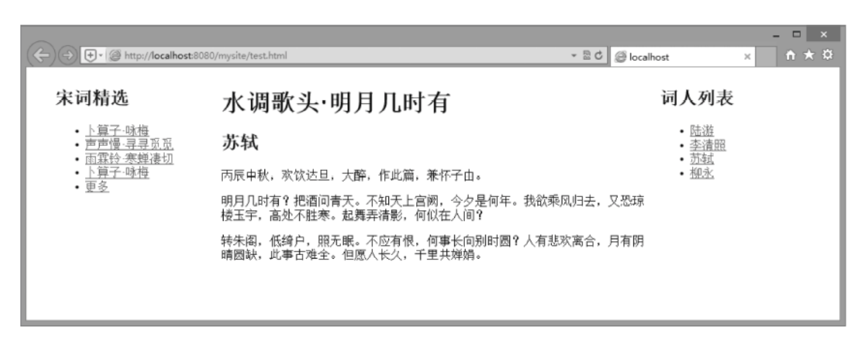


图 13.2　显示屏幕宽度在 999 像素以上时页面显示效果

当显示屏幕宽度在 639 像素以上、1000 像素以下时，两栏显示，预览效果如图 13.3 所示；当显示屏幕宽度在 640 像素以下时，3 个栏目从上往下堆叠显示，预览效果如图 13.4 所示。



图 13.3　宽度在 639 像素以上、1000 像素以下时效果

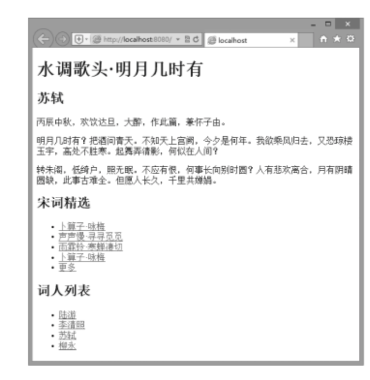


图 13.4　宽度在 640 像素以下时效果



13.2.3 设计响应式菜单

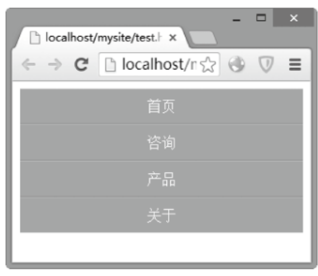


Note



视频讲解

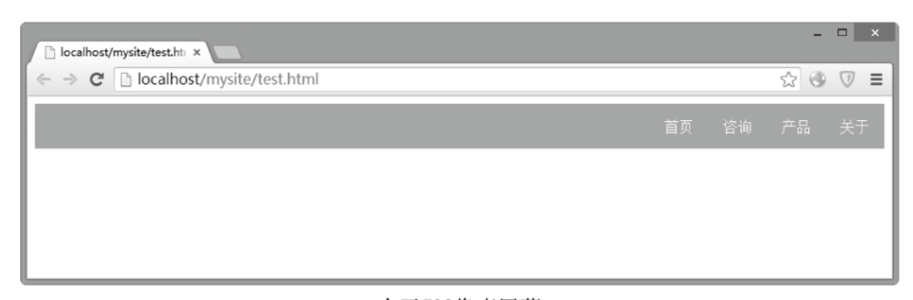
本例设计一个响应式菜单，能够根据设备显示不同的伸缩盒布局效果。在小屏设备上，从上到下显示；在默认状态下，从左到右显示，右对齐盒子；当设备小于 801 像素时，设计导航项目分散对齐显示，预览效果如图 13.5 所示。



小于601像素屏幕



介于600和800像素之间设备



大于799像素屏幕

图 13.5　定义伸缩项目居中显示

示例主要代码如下。

```
<style type="text/css">
/*默认伸缩布局*/
.navigation {
    list-style: none;
    margin: 0;
    background: deepskyblue;
    /* 启动伸缩盒布局 */
    display: -webkit-box;
    display: -moz-box;
    display: -ms-flexbox;
    display: -webkit-flex;
    display: flex;
    -webkit-flex-flow: row wrap;
    /* 所有列面向主轴终点位置靠齐 */
```






```
        justify-content: flex-end;
}
/*设计导航条内超链接默认样式*/
.navigation a { text-decoration: none; display: block; padding: 1em; color: white;}
/*设计导航条内超链接在鼠标指针经过时的样式*/
.navigation a:hover { background: blue; }
/*在小于 801 像素设备下伸缩布局*/
@media all and (max-width: 800px) {
    /* 当在中等屏幕中，导航项目居中显示，并且剩余空间平均分布在列表之间 */
    .navigation { justify-content: space-around; }}
/*在小于 601 像素设备下伸缩布局*/
@media all and (max-width: 600px) {
    .navigation { /* 在小屏幕下，没有足够空间进行行排列，可以换成列排列 *
        -webkit-flex-flow: column wrap;
        flex-flow: column wrap;
        padding: 0;}
    .navigation a {
        text-align: center;
        padding: 10px;
        border-top: 1px solid rgba(255,255,255,0.3);
        border-bottom: 1px solid rgba(0,0,0,0.1);}
    .navigation li:last-of-type a { border-bottom: none; }
}
</style>

<ul class="navigation">
    <li><a href="#">首页</a></li>
    <li><a href="#">咨询</a></li>
    <li><a href="#">产品</a></li>
    <li><a href="#">关于</a></li>
</ul>
```

13.2.4　设计自动隐藏布局



本例将设计一个初步响应式页面布局效果，并能根据显示屏幕宽度变化自动隐藏或调整版式显示。

【操作步骤】

第 1 步，新建 HTML5 文档，在头部<head>标签内定义视口信息。使用<meta>标签设置视口缩放比例为 1，让浏览器使用设备的宽度作为视图的宽度，并禁止初始缩放，代码如下。






```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
```

第 2 步，IE 8 或者更早的浏览器并不支持媒体查询。可以使用 media-queries.js 或者 respond.js 插件进行兼容，代码如下。

```
<!--[if lt IE 9]>
    <script src="http://css3-mediaqueries-js.googlecode.com/svn/trunk/css3-mediaqueries.js"></script>
<![endif]-->
```

第 3 步，设计页面 HTML 结构。整个页面基本布局包括头部、内容、侧边栏和页脚。内容容器宽度是 600 像素，而侧边栏宽度是 300 像素。代码如下，效果如图 13.6 所示。

```
<div id="pagewrap">
    <div id="header">
        <h1>唐诗赏析</h1>
    </div>
    <div id="content">
        <h1>水调歌头·明月几时有</h1>
        <h2>苏轼</h2>
        <p>……</p>
    </div>
    <div id="sidebar">
        <h2>宋词精选</h2>
        <ul>
            <li>……</li>
        </ul>
    </div>
    <div id="footer">
        <h2>词人列表</h2>
        <ul>
            <li>……</li>
        </ul>
    </div>
</div>
```





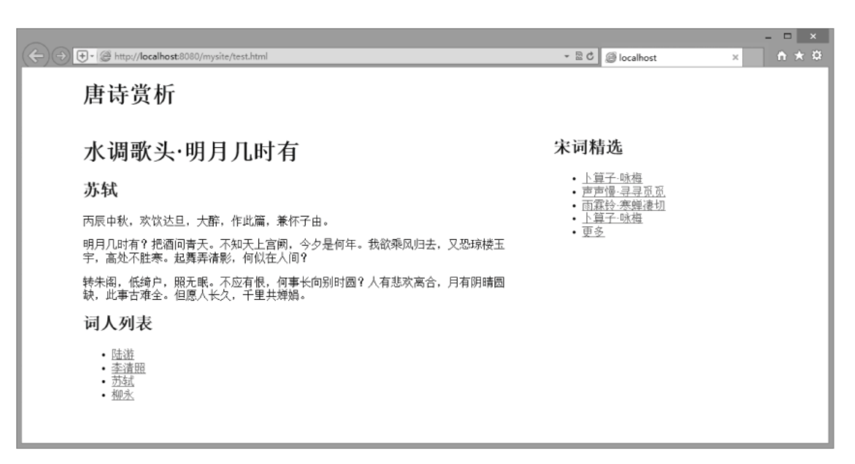


图 13.6　设计页面结构

第 4 步，使用 CSS3 媒体查询设计当视图宽度小于等于 980 像素时，如下规则将会生效。基本上，会将所有的容器宽度从像素值设置为百分比以使得容器大小自适应，代码如下。

```
/* 当窗口视图小于等于 980 像素时响应下面样式 */
@media screen and (max-width: 980px) {
    #pagewrap { width: 94%; }
    #content { width: 65%; }
    #sidebar { width: 30%; }
}
```

第 5 步，为小于等于 700 像素的视图指定<div id="content">和<div id="sidebar">的宽度为自适应，并且清除浮动，使得这些容器按全宽度显示，代码如下。

```
/* 当窗口视图小于等于 700 像素时响应下面样式 */
@media screen and (max-width: 700px) {
    #content {
        width: auto;
        float: none;
    }
    #sidebar {
        width: auto;
        float: none;
    }
}
```

第 6 步，对于小于等于 480 像素（手机屏幕）的情况，将 h1 和 h2 的字体大小修改为 16 像素，并隐藏侧边栏<div id="sidebar">，代码如下。

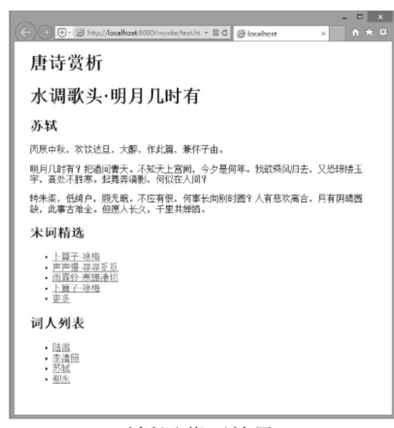




```
/* 当窗口视图小于等于 480 像素时响应下面样式 */
@media screen and (max-width: 480px) {
    h1, h2 { font-size: 16px; }
    #sidebar { display: none; }
}
```

Note

第 7 步，可以根据需要添加更多媒体查询，目的在于为指定的视图宽度指定不同的 CSS 规则，从而实现不同的布局。示例演示效果如图 13.7 所示。



平板屏幕下效果



手机屏幕下效果

图 13.7　设计不同宽度下的视图效果



视频讲解

13.2.5　设计自适应手机页面

本例设计页面宽度为 980 像素，对于桌面屏幕来说，该宽度适用于任何宽于 1024 像素的分辨率。通过媒体查询监测宽度小于 980 像素的设备，并将页面宽度由固定方式改为液态版式，布局元素的宽度随着浏览器窗口的尺寸变化进行调整。当可视部分的宽度进一步减小到 650 像素以下时，主要内容部分的容器宽度会增大至全屏，而侧边栏将被置于主内容部分的下方，整个页面变为单列布局。演示效果如图 13.8 所示。





图 13.8　在不同宽度下的视图效果





【操作步骤】

第 1 步，新建 HTML5 文档，构建文档结构，包括页头、主要内容部分、侧边栏和页脚，代码如下。

```
<div id="pagewrap">
    <header id="header">
        <hgroup>
            <h1 id="site-logo">网站 LOGO</h1>
            <h2 id="site-description">网站描述信息</h2>
        </hgroup>
        <nav>
            <ul id="main-nav">
                <li><a href="#">导航链接，可以扩展</a></li>
            </ul>
        </nav>
        <form id="searchform">
            <input type="search">
        </form>
    </header>
    <div id="content">
        <article class="post">主体内容区域</article>
    </div>
    <aside id="sidebar">
        <section class="widget"> 侧栏栏目</section>
    </aside>
    <footer id="footer">页脚区域</footer>
</div>
```

第 2 步，IE 9 之前的浏览器不支持 HTML5 标签，使用 html5.js 来帮助这些旧版本的 IE 浏览器创建 HTML5 元素节点，代码如下。

```
<!--[if lt IE 9]>
<script src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
<![endif]-->
```

第 3 步，设计 HTML5 块级元素样式，将这些新元素声明为块级样式，代码如下。

```
article, aside, details, figcaption, figure, footer, header, hgroup, menu, nav, section {display: block; }
```

第 4 步，设计主要结构的 CSS 样式。这里将注意力集中在整体布局上。整体设计在默认情况下页面容器的固定宽度为 980 像素，页头部分（header）的固定高度为 160 像素，主要内容部分（content）





的宽度为 600 像素，左浮动。侧边栏（sidebar）右浮动，宽度为 280 像素，代码如下。

```
<style type="text/css">
#pagewrap {
    width: 980px;
    margin: 0 auto;
}
#header { height: 160px; }
#content {
    width: 600px;
    float: left;
}
#sidebar {
    width: 280px;
    float: right;
}
#footer { clear: both; }
</style>
```

第 5 步，在页面中调用 css3-mediaqueries.js 文件，解决 IE 8 及以前版本支持 CSS3 媒体查询的问题，代码如下。

```
<!--[if lt IE 9]>
    <script src="http://css3-mediaqueries-js.googlecode.com/svn/trunk/css3-mediaqueries.js"></script>
<![endif]-->
```

第 6 步，创建 CSS 样式表，并在页面中调用，代码如下。

```
<link href="media-queries.css" rel="stylesheet" type="text/css">
```

第 7 步，借助媒体查询设计自适应布局。

当浏览器可视部分宽度大于 650 像素、小于 981 像素时，将 pagewrap 的宽度设置为 95%，将 content 的宽度设置为 60%，将 sidebar 的宽度设置为 30%，代码如下。

```
@media screen and (max-width: 980px) {
    #pagewrap { width: 95%; }
    #content {
        width: 60%;
        padding: 3% 4%;
    }
    #sidebar { width: 30%; }
```






```
    #sidebar .widget {
        padding: 8% 7%;
        margin-bottom: 10px;
    }
}
```

第 8 步，当浏览器可视部分宽度小于 651 像素时，将 header 的高度设置为 auto；将 searchform 绝对定位在 top: 5px 的位置；将 main-nav、site-logo、site-description 的定位设置为 static；将 content 的宽度设置为 auto（主要内容部分的宽度将扩展至满屏），并取消 float 设置；将 sidebar 的宽度设置为 100%，并取消 float 设置，代码如下。

```
@media screen and (max-width: 650px) {
    #header { height: auto; }
    #searchform {
        position: absolute;
        top: 5px;
        right: 0;
    }
    #main-nav { position: static; }
    #site-logo {
        margin: 15px 100px 5px 0;
        position: static;
    }
    #site-description {
        margin: 0 0 15px;
        position: static;
    }
    #content {
        width: auto; margin: 20px 0;
        float: none;
    }
    #sidebar {
        width: 100%; margin: 0;
        float: none;
    }
}
```

第 9 步，当浏览器可视部分宽度小于 481 像素时，480 像素也就是传统手机横屏时的宽度。当可视部分的宽度小于 481 像素时，禁用 HTML 节点的字号自动调整。默认情况下，手机会将过小的字号放





大，这里可以通过-webkit-text-size-adjust 属性进行调整，将 main-nav 中的字号设置为 90%，代码如下。

```
@media screen and (max-width: 480px) {
    html {-webkit-text-size-adjust: none;}
    #main-nav a {
        font-size: 90%;
        padding: 10px 8px;
    }
}
```

第 10 步，设计弹性图片。为图片设置 max-width: 100%和 height: auto，设计图像弹性显示，代码如下。

```
img {
    max-width: 100%; height: auto;
    width: auto\9; /*兼容 IE 8 */
}
```

第 11 步，设计弹性视频。对于视频也需要做 max-width: 100%的设置，但是 Safari 对 embed 的该属性支持不是很好，所以使用 width: 100%来代替，代码如下。

```
.video embed, .video object, .video iframe {
    width: 100%; min-height: 300px;
    height: auto;
}
```

第 12 步，在默认情况下，手机端 Safari 浏览器会对页面进行自动缩放，以适应屏幕尺寸。这里可以使用以下的 meta 设置，将设备的默认宽度作为页面在 Safari 的可视部分宽度，并禁止初始化缩放，代码如下。

```
<meta name="viewport" content="width=device-width; initial-scale=1.0">
```

13.3 在线练习

1. 练习响应式网页设计。
2. 复习 CSS3 重要属性，强化训练和应用 CSS3 新功能的能力。



在线练习 1



在线练习 2



使用 JavaScript 控制 CSS 样式

在网页设计中，经常会用 JavaScript 代码控制页面样式，这种样式也称为脚本样式，因此用户要了解 JavaScript 代码的基本用法和操作 CSS 样式的一般方法。本章将介绍如何使用 JavaScript 控制 CSS 样式来设计各种动态效果。

【学习要点】

- 了解使用 JavaScript 控制 CSS 样式的方法。
- 使用 JavaScript 控制网页对象的大小和显隐。
- 设计运动效果。
- 设计渐隐、渐显效果。
- 在网页中添加各种交互式响应或动态特效。





Note



视频讲解

14.1 在网页中使用 JavaScript 脚本

JavaScript 是目前最流行、应用最广泛的 Web 编程语言。一般情况下，JavaScript 代码只能够在网页中发挥作用，当然编写 JavaScript 代码的方法也很简单。

14.1.1 使用<script>标签

在 HTML 页面中嵌入 JavaScript 脚本需要使用<script>标签，用户可以在<script>标签中直接编写 JavaScript 代码，或者单独编写 JavaScript 文件，然后通过<script>标签导入。

【示例 1】 直接在页面中嵌入 JavaScript 代码。

第 1 步，新建 HTML 文档，保存为 test.html。然后在<head>标签内插入一个<script>标签。

第 2 步，为<script>标签指定 type 属性值为"text/javascript"。现代浏览器默认<script>标签的类型为 JavaScript 脚本，因此省略 type 属性，依然能够被正确执行，但是考虑到代码的兼容性，建议定义该属性。

第 3 步，直接在<script>标签内部输入以下 JavaScript 代码。

```
<script type="text/javascript">
function hi(){
    document.write("<h1>Hello,World!</h1>");
}
hi();
</script>
```

上面 JavaScript 脚本先定义了一个 hi()函数，该函数被调用后会在页面中显示字符"Hello,World! "。document 表示 DOM 网页文档对象，document.write()表示调用 Document 对象的 write()方法，在当前网页源代码中写入 HTML 字符串"<h1>Hello,World!</h1>"。

调用 hi()函数，浏览器将在页面中显示一级标题字符"Hello,World! "。

第 4 步，保存网页文档，在浏览器中预览，显示效果如图 14.1 所示。



图 14.1　第一个 JavaScript 程序

包含在<script>标签内的 JavaScript 代码被浏览器从上至下依次解释执行。





【示例 2】包含外部 JavaScript 文件。

第 1 步，新建文本文件，保存为 test.js。注意，扩展名为.js，表示该文本文件是 JavaScript 类型的文件。

提示：使用<script>标签包含外部 JavaScript 文件时，默认文件类型为 JavaScript，因此扩展名.js 不是必需的，浏览器不会检查包含 JavaScript 的文件的扩展名。在高级开发中，使用 JSP、PHP 或其他服务器端语言动态生成 JavaScript 代码时可以使用任意扩展名，如果不使用.js 扩展名，用户应确保服务器能返回正确的 MIME 类型。

第 2 步，打开 test.js 文本文件，在其中编写下面的代码，定义简单的输出函数。

```
function hi(){
    alert("Hello,World!");
}
```

在上面的代码中，alert()表示 Window 对象的方法，调用该方法将弹出一个提示对话框，显示参数字符串"Hello,World!"。

第 3 步，保存 JavaScript 文件，注意与网页文件的位置关系。这里保存 JavaScript 文件位置与调用该文件的网页文件位于相同目录下。

第 4 步，新建 HTML 文档，保存为 test1.html。然后在<head>标签内插入一个<script>标签。定义 src 属性，设置属性值为指向外部 JavaScript 文件的 URL 字符串。代码如下。

```
<script type="text/javascript" src="test.js"></script>
```

第 5 步，在上面<script>标签下一行继续插入一个<script>标签，直接在<script>标签内部输入 JavaScript 代码，调用外部 JavaScript 文件中的 hi()函数。

```
<script type="text/javascript" src="test.js"></script>
<script type="text/javascript">
hi();            //调用外部 JavaScript 文件的函数
</script>
```

第 6 步，保存网页文档，在浏览器中预览，显示效果如图 14.2 所示。



图 14.2　调用外部函数弹出提示对话框





Note


视频讲解

提示：定义 src 属性的<script>标签不应再包含 JavaScript 代码。如果嵌入了代码，则只会下载并执行外部 JavaScript 文件，嵌入代码会被忽略。src 属性可以包含外部域的 JavaScript 文件，如下所示。

```
<script type="text/javascript" src="http://www.sothersite.com/test.js"></script>
```

14.1.2 比较脚本样式与 CSS 样式

JavaScript 代码与 CSS 代码不会相互干扰，但是由于 JavaScript 可以控制 CSS 样式，所以它们之间仍然存在某些关联。对于 CSS 文件来说，样式所引用的外部文件的路径都是以代码所在位置作为参考来进行设置的，而 JavaScript 恰恰相反，它是以所引用的网页位置作为参考进行设置的。

【示例】有一个简单的站点结构，网页文件位于根目录，而 CSS 文件、JavaScript 文件和图像文件都位于根目录下 images 文件夹中，如图 14.3 所示。

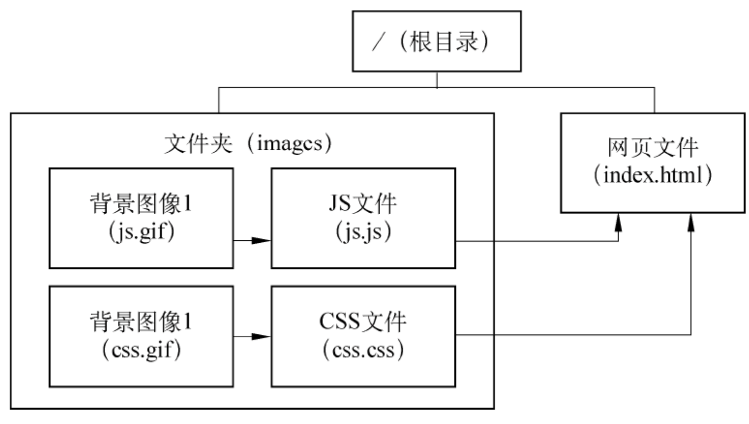


图 14.3　一个简单站点结构

下面分别使用 CSS 样式和 JavaScript 脚本样式为网页中的<div id="box">标签定义背景图像。

【操作步骤】

第 1 步，新建样式表文件，保存为 css.css，存放于 images 文件夹中。

第 2 步，在 CSS 样式表文件（css.css）中定义方法如下。

```
#box {
    background:url(css.gif);
}
```

CSS 文件与背景图像文件都在同一目录（images 文件夹）下，所以可以直接引用，而不用考虑网页文件的位置。

第 3 步，新建 JavaScript 文本，保存为 js.js，存放于 images 文件夹中。

第 4 步，在 js.js 文件中输入下面的代码，使用 JavaScript 脚本定义<div id="box">的背景图像。





```
window.onload = function(){
    document.getElementById("box").style.backgroundImage="url(images/ js.gif)";
}
```

从上面的代码可以看到，JavaScript 文件所引用的背景图像路径是以网页文件的位置为参考来进行设置的，而不用考虑 JavaScript 文件的具体位置，如果网页文件不动，则 JavaScript 文件所引用的路径是不会变化的。

第 5 步，新建网页文件，保存为 index.html，存放于根目录下。

第 6 步，在网页文件中同时引用 CSS 和 JavaScript 文件，代码如下。

```
<style type="text/css">
#box {
    width:440px;
    height:312px;
}
</style>
<script type="text/javascript"   src="images/js.js"></script>
<link href="images/css.css" rel="stylesheet" type="text/css">

<div id="box"></div>
```

第 7 步，保存网页文档，在浏览器中预览，会发现<div id="box">标签显示 JavaScript 脚本定义的背景图像效果，如图 14.4 所示。

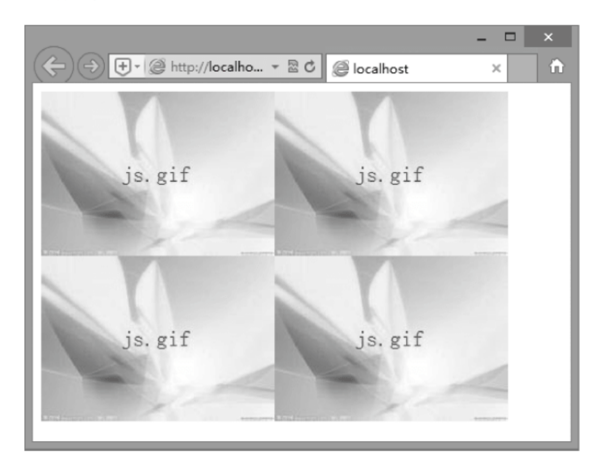


图 14.4　js.gif 优先显示

总之，JavaScript 文件与 CSS 文件中代码在引用外部图像文件时，它们的相对路径设置是不同的，具体区分如下。

☑　CSS 文件：考虑 CSS 文件与导入的外部图像文件之间的位置关系。



☑ JavaScript 文件：考虑网页文件与导入的外部图像文件之间的位置关系。

同时，当使用 CSS 和 JavaScript 同时为页面对象定义样式时，JavaScript 脚本样式的优先级要大于 CSS 样式的优先级。



14.2 获取网页对象

使用 JavaScript 控制 CSS 样式的第一步是需要获取网页对象，以实现对其控制。



14.2.1 获取元素

为了获取文档结构中的元素节点，DOM 提供了以下两个方法。

1. *使用 getElementById()方法*

使用 getElementById()方法可以精确获取指定元素的引用指针，具体用法如下。

```
o = document.getElementById(ID)
```

其中 o 表示指定元素的引用指针，参数 ID 表示文档结构中对应元素的 id 属性值。如果文档中不存在指定元素，则返回值为 null。该方法只适用于 document 对象。

【示例 1】下面的脚本能够获取对<div id="box">对象的控制权。

```
<div id="box">盒子</div>
<script>
var box = document.getElementById("box");   // 获取 id 属性值为 box 的指定元素的引用指针
</script>
```

getElementById()方法返回指定元素的对象，这个对象包含 nodeName、nodeType 等属性，简单说明如下。

☑ nodeName 表示节点的名称。如果是元素节点，则 nodeName 返回值为标签名称，标签名称永远是大写；如果是属性节点，则 nodeName 返回值为属性的名称；如果是文本节点，则 nodeName 返回值永远是#text 标识符；如果是文档节点，则 nodeName 返回值永远是#document 标识符。

☑ nodeType 表示节点的类型。该属性的返回值比较多，常用节点类型包括：1 表示元素类型，2 表示属性，3 表示文本，8 表示注释，9 表示文档。

【示例 2】在本示例中，使用 getElementById()方法获取<div id="box">对象的引用指针，然后利用 nodeName、nodeType 属性查看该对象的节点类型和节点名称，代码如下。

```
<div id="box">盒子</div>
<script>
var box = document.getElementById("box");          // 获取指定盒子的引用指针
```



```
var info = "nodeName： " + box.nodeName;            // 获取该节点的名称
info += "\rnodeType： " + box.nodeType;             // 获取该节点的类型
alert(info);                                        // 显示提示信息
</script>
```





2. 使用 getElementByTagName()方法

使用 getElementByTagName()方法获取指定标签名称的所有元素对象，其用法如下。

```
a = document.getElementsByTagName(tagName)
```

其中参数 tagName 表示指定名称的标签，该方法返回值为一个元素集合。使用 length 属性可以获取集合中包含元素的个数，利用数组下标可以确定其中某个元素对象。

【示例 3】对于这些数组元素来说，由于它们都是节点对象，因此可以使用 nodeName、nodeType 属性查看该对象的节点类型和节点名称，代码如下。

```
<p id="p1">段落文本 1</p>
<p id="p2">段落文本 2</p>
<p id="p3">段落文本 3</p>
<script>
var p = document.getElementsByTagName("p");          // 获取文档中所有 p 元素
alert(p[2].nodeName);                                // 显示第 3 个 p 元素对象的节点名称
</script>
```

在实际开发中，常用 for 循环遍历集合中所有元素。

【示例 4】下面的代码就是使用 for 结构遍历获得的所有 p 元素，并设置 p 元素的 class 属性为 red。

```
<p id="p1">段落文本 1</p>
<p id="p2">段落文本 2</p>
<p id="p3">段落文本 3</p>
<script>
var p = document.getElementsByTagName("p");     // 获取文档中所有 p 元素
for(var i=0;i<p.length;i++){                     // 遍历 p 数据集合
    p[i].setAttribute("class","red");            // 为每个 p 元素添加 class 类
}
</script>
```

提示：使用 document.getElementsByTagName("*")方式可获取文档中所有元素节点。不过这个方法很少用，同时 IE 6.0 及以下版本浏览器对其支持不是很好。对于 IE 浏览器来说，可以通过 document.all 来获取文档中所有元素的节点。



14.2.2 使用 CSS 选择器匹配元素



Note



视频讲解

HTML5 引入了与 jQuery 选择器相似的 DOM API 模块，该模块中的 querySelector()和 querySelectorAll() 方法能够根据 CSS 选择器规范，便捷定位文档中指定的元素。目前主流浏览器均支持它们，包括 IE8+、Firefox、Chrome、Safari 和 Opera。

querySelector()和 querySelectorAll()方法的参数必须是符合 CSS 选择器规范的字符串。不同的是，querySelector()方法返回的是一个元素对象，querySelectorAll()方法返回的是一个元素集合。

【示例 1】新建网页文档，输入下面的 HTML 结构代码。

```
<div class="content">
    <ul>
        <li>首页</li>
        <li class="red">财经</li>
        <li class="blue">娱乐</li>
        <li class="red">时尚</li>
        <li class="blue">互联网</li>
    </ul>
</div>
```

如果要获得第 1 个 li 元素，可以使用如下方法。

```
document.querySelector(".content ul li");
```

如果要获得所有 li 元素，可以使用如下方法。

```
document.querySelectorAll(".content ul li");
```

如果要获得所有 class 为 red 的 li 元素，可以使用如下方法。

```
document.querySelectorAll("li.red");
```

提示： DOM API 模块也包含 getElementsByClassName()方法，使用该方法可以获取指定类名的元素。例如：

```
document.getElementsByClassName("red");
```

注意，getElementsByClassName()方法只能够接收字符串，且为类名，而不需要加点号前缀，如果没有匹配到任何元素则返回空数组。

CSS 选择器是一个便捷的确定元素的方法，这是因为大家已经对 CSS 很熟悉了。当需要联合查询时，使用 querySelectorAll()更加便利。





【示例 2】在文档中一些 li 元素的 class 名称是 red，另一些 class 名称是 blue，可以用 querySelectorAll()方法一次性获得这两类节点，代码如下。

```
var lis =   document.querySelectorAll("li.red, li.blue");
```

如果不使用 querySelectorAll()方法，那么要获得同样的列表，需要做更多工作。一个办法是选择所有的 li 元素，然后通过迭代操作过滤出那些不需要的列表项目，代码如下。

```
var result = [], lis1 = document.getElementsByTagName('li'), classname = '';
for(var i = 0, len = lis1.length; i < len; i++) {
    classname = lis1[i].className;
    if(classname === 'red' || classname === 'blue') {
        result.push(lis1[i]);
    }
}
```

比较上面两种不同的用法，使用选择器 querySelectorAll()方法比使用 getElementsByTagName()的性能要快很多。因此，如果浏览器支持 document.querySelectorAll()，那么最好使用它。

14.3　操作类样式

使用 JavaScript 控制 CSS 样式最简单、最直接的方法是为元素添加或删除类样式。

14.3.1　获取类样式



DOM 定义 getAttribute()方法可以获取指定元素的属性。其用法比较简单，只要指定元素及它的属性，即可快速反馈该元素所对应的属性值。

【示例 1】本示例能够获取红色盒子和蓝色盒子，并显示这些元素所包含的 class 属性值，代码如下。

```
<script>
window.onload = function() {
    var red = document.getElementById("red");           // 获取红色盒子
    alert(red.getAttribute("class"));                   // 显示红色盒子的 class 属性值
    var blue = document.getElementById("blue");         // 获取蓝色盒子
    alert(blue.getAttribute("class"));                  // 显示蓝色盒子的 class 属性值
}
</script>
<div id="red" class="red">红盒子</div>
<div id="blue" class="blue">蓝盒子</div
```



Note

所传递的参数是一个字符串形式的元素属性名称。返回的是一个字符串类型的值，如果给定属性不存在，则返回的值为 null。

【示例 2】除了上面标准读取属性的方法外，HTML DOM 模型还支持快捷读取属性的方法，代码如下。

```
window.onload = function() {
    var red = document.getElementById("red");
    alert(red.id);
    var blue = document.getElementById("blue");
    alert(blue.id);
}
```

但是对于 class 属性，则必须使用 className 属性来读取，因为 class 是 JavaScript 保留字。同样，要读取 for 属性，则必须使用 htmlFor 属性名，这与 CSS 脚本中 float 和 text 属性被改名为 cssFloat 和 cssText 原因相同。

【示例 3】使用 className 读取类样式，代码如下。

```
<script>
window.onload = function() {
    var red = document.getElementById("red");          // 获取红色盒子
    alert(red.className);                              // 显示红色盒子的 class 属性值
    var blue = document.getElementById("blue");        // 获取蓝色盒子
    alert(blue.className);                             // 显示蓝色盒子的 class 属性值
}
</script>
<div id="red" class="red">红盒子</div>
<div id="blue" class="blue">蓝盒子</div
```

【示例 4】对于复合类样式，需要使用 split()方法劈开返回字符串，然后遍历读取类样式，代码如下。

```
<script>
window.onload = function() {
    // 所有类名生成的数组
    var classNameArray = document.getElementById("red").className.split(" ");
    for(var i in classNameArray ){        // 遍历数组
        alert(classNameArray[i]);         // 当前 class 名
    }
}
</script>
```



```
<div id="red" class="red blue">红盒子</div>
```

14.3.2　添加类样式

为元素设置属性可以使用 setAttribute()方法实现，用法如下。

```
e.setAttribute(name,value)
```

参数 e 表示指定的元素对象，参数 name 和 value 分别表示属性名称和属性值。属性名称和属性值必须以字符串的形式进行传递。如果元素中存在指定的属性，则它的值将被刷新；如果不存在，则 setAttribute()方法将为元素创建该属性并赋值。

【示例 1】本示例分别为页面中的 div 元素设置 class 属性，代码如下。

```
<script>
window.onload = function() {
    var red = document.getElementById("red");
    var blue = document.getElementById("blue");
    red.setAttribute("class", "red");
    blue.setAttribute("class", "blue");
}
</script>
<div id="red">红盒子</div>
<div id="blue">蓝盒子</div>
```

【示例 2】使用 setAttribute()方法存在弊端，一般通过 className 设置元素的类名，代码如下。

```
<script>
window.onload = function() {
    var red = document.getElementById("red");
    var blue = document.getElementById("blue");
    red.className = "red";
    blue.className = "blue";
}
</script>
<div id="red">红盒子</div>
<div id="blue">蓝盒子</div>
```

【示例 3】直接使用 className 添加类样式，会覆盖元素原来的类样式。我们可以采用叠加的方式添加类，代码如下。

```
<script>
window.onload = function() {
```








Note

```
        var red = document.getElementById("red");
        red.className = "red";
        red.className +=    " blue";
    }
    </script>
    <div id="red">红盒子</div>
```

【示例 4】使用叠加的方式添加类也存在问题，这样容易添加大量重复的类。为此，我们定义一个检测函数，判断元素是否包含指定的类，然后决定是否添加类，代码如下。

```
<script>
function hasClass(element,className){                //类名检测函数
    var reg =new RegExp('(\\s|^)'+ className + '(\\s|$)');
    return    reg.test(element.className);            //使用正则检测是否有相同的样式
}
function addClass(element,className){                //添加类名函数
    if(!hasClass(element, className))
        element.className +=' ' + className;
}
window.onload = function() {
    var red = document.getElementById("red");
    addClass(red,'red');
    addClass(red,'blue');
}
</script>
<div id="red">红盒子</div>
```

14.3.3　删除类样式


视频讲解

DOM 使用 removeAttribute()方法可以删除指定的属性，用法如下。

```
e.removeAttribute(name)
```

其中 e 表示一个元素对象，而参数 name 表示元素的属性名。

【示例 1】本示例演示了如何动态设置表格的边框，代码如下。

```
<script>
window.onload = function() {// 绑定页面加载完毕时的事件处理函数
    var table = document.getElementsByTagName("table")[0];// 获取表格外框的引用指针
    var del = document.getElementById("del"); // 获取删除按钮的引用指针
    var reset = document.getElementById("reset");            // 获取恢复按钮的引用指针
```





```
        del.onclick = function(){                        // 为删除按钮绑定事件处理函数
            table.removeAttribute("border");             // 移出边框属性
        }
        reset.onclick = function(){                      // 为恢复按钮绑定事件处理函数
            table.setAttribute("border", "2");           // 设置表格的边框属性
        }
    }
    </script>
    <table width="100%" border="2">
        <tr>
            <td>数据表格</td>
        </tr>
    </table>
    <button id="del">删除</button><button id="reset">恢复</button>
```

在上面的示例中，设计了两个按钮，并分别绑定不同的事件处理函数。单击【删除】按钮即可调用表格的 removeAttribute()方法清除表格边框，单击【恢复】按钮即可调用表格的 setAttribute()方法重新设置表格边框的粗细。

【示例 2】本示例演示了如何自定义删除类函数，并调用该函数删除指定类名，代码如下。

```
<script type="text/javascript">
function hasClass(element,className){//类名检测函数
    var reg =new RegExp('(\\s|^)'+ className + '(\\s|$)');
    return   reg.test(element.className); //使用正则检测是否有相同的样式
}function deleteClass(element,className){
     f(hasClass(element,className)){
        element.className.replace(reg,' '); //利用正则捕获到要删除的样式的名称，然后把它替换成一个空
                                            白字符串，就相当于将其删除
    }
}
window.onload = function() {
    var red = document.getElementById("red");
    deleteClass(red,'blue');
}
</script>
<div id="red" class="red   blue   bold">红盒子</div>
```

上面的代码是使用正则表达式检测 className 属性值字符串中是否包含指定的类名，如果存在，则使用空字符替换匹配到的子字符串，从而实现删除类名的目的。






视频讲解

14.4　操作 CSS 样式

在 JavaScript 脚本中获取页面元素之后，就可以使用 style 属性获取该元素的 CSS2Properties 对象。CSS2Properties 包含了该对象的所有 CSS 脚本属性。设置这些属性与设置 CSS 样式的效果是一样的。

14.4.1　使用 style 对象

DOM 定义每个元素都继承一个 style 对象，style 对象包含一些方法，利用这些方法可以与 CSS 样式实现交互。但是，style 对象针对的是行内样式，不支持操作样式表，包括内部样式表（<style> 标签包含的样式）和外部样式表。

1. getPropertyValue()*方法*

getPropertyValue()能够获取指定元素样式属性的值。具体用法如下。

```
var value = e.style.getPropertyValue(propertyName)
```

参数 propertyName 表示 CSS 属性名，不是 CSS 脚本属性名，对于复合名应该使用连字符进行连接。

【示例 1】下面的代码使用 getPropertyValue()方法获取行内样式中的 width 属性值，然后输出到盒子内显示，如图 14.5 所示。

```
<script>
window.onload = function(){
    var box = document.getElementById("box");          //获取<div id="box">
    var width = box.style.getPropertyValue("width");   //读取 div 元素的 width 属性值
    box.innerHTML =  "盒子宽度：" + width;              //输出显示 width 值
}
</script>

<div id="box" style="width:300px; height:200px;border:solid 1px red" >盒子</div>
```

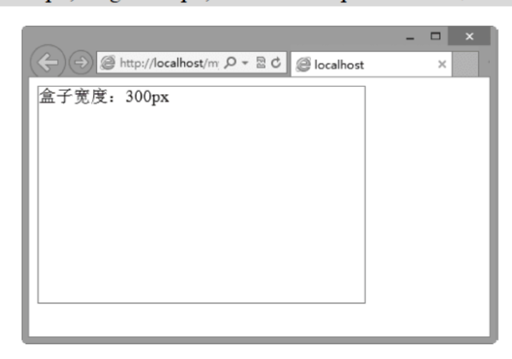


图 14.5　使用 getPropertyValue()读取行内样式







早期 IE 版本不支持 getPropertyValue()方法，但是可以通过 style 对象直接访问样式属性，以获取指定样式的属性值。

提示：在设置 CSS 脚本属性时，应注意以下几个问题。

- 由于 float 是 JavaScript 保留字，禁止用户使用。因此，CSS2Properties 内没有与 float 属性对应的名称。为了解决这个问题，CSS2Properties 在 float 属性前增加了 css 前缀，使用 cssFloat 名来表示脚本中的 float 属性。
- 在 CSS 中读写属性值时，不需要考虑值的类型。但是在 JavaScript 中，CSS2Properties 对象认定所有 CSS 属性值都是字符串，因此脚本中所有属性值都必须加上引号，以表示为字符串数据类型。例如：

```
elementNode.style.fontFamily = "Arial, Helvetica, sans-serif";
elementNode.style.cssFloat = "left";
elementNode.style.color = "#ff0000";
```

- 在 CSS 样式中声明尾部的分号不能够作为属性值的一部分被引用，脚本中的分号只是 JavaScript 语法规则的一部分，而不是 CSS 声明中分号的引用。
- 声明中属性值所包含的单位等都必须作为值的一部分，完整地传递给 CSS 脚本属性，省略单位则所设置的脚本样式无效。例如：

```
elementNode.style.width = "100px";
```

- 在脚本中可以动态设置属性值，但最终赋值给属性的值应是一个字符串，且必须包含单位。例如：

```
elementNode.style.top = top + "px";
elementNode.style.right = right + "px";
elementNode.style.bottom = bottom + "px";
elementNode.style.left = left + "px";
```

【示例 2】针对上面的示例代码，可以使用如下方式读取 width 属性值。

```
<script>
window.onload = function(){
    var box = document.getElementById("box");
    var width = box.style.width;
    box.innerHTML = "盒子宽度：" + width;
}
</script>
```



2. setProperty()方法



setProperty()方法为指定元素设置样式，具体用法如下。

```
e.style.setProperty(propertyName, value, priority)
```

参数说明如下。

☑ propertyName：设置 CSS 属性名。
☑ value ：设置 CSS 属性值，包含属性值的单位。
☑ priority：表示是否设置!important 优先级命令，如果不设置可以以空字符串表示。

【示例 3】在本示例中使用 setProperty()方法定义盒子的显示宽度和高度分别为 400 像素和 200 像素，代码如下。

```
<script>
window.onload = function(){
    var box = document.getElementById("box");    //获取<div id="box">
    box.style.setProperty("width","400px","");    //定义盒子宽度为 400 像素
    box.style.setProperty("height","200px","");   //定义盒子高度为 200 像素
}
</script>

<div id="box" style="border:solid 1px red" >盒子</div>
```

如果要兼容早期 IE 浏览器，则可以使用如下方式设置。

```
<script>
window.onload = function(){
    var box = document.getElementById("box");
    box.style.width = "400px";
    box.style.height = "200px";
}
</script>
```

3. removeProperty()方法

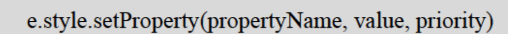
removeProperty()方法可以移出指定 CSS 属性的样式声明，具体用法如下。

```
e.style. removeProperty (propertyName)
```

4. item()方法

item()方法返回 style 对象中指定索引位置的 CSS 属性名称，具体用法如下。

```
var name = e.style.item(index)
```





参数 index 表示 CSS 样式的索引号。

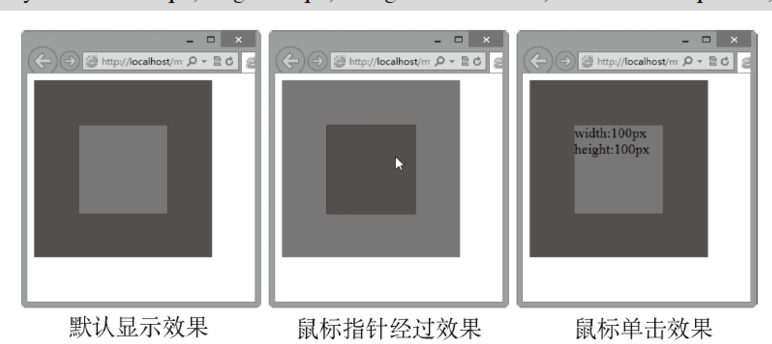
5. getPropertyPriority()方法

getPropertyPriority()方法可以获取指定 CSS 属性中是否附加了!important 优先级命令，如果存在，则返回 important 字符串，否则返回空字符串。

【示例 4】在本示例中，定义鼠标指针移过盒子时，设置盒子的背景色为蓝色，而边框颜色为红色，当移出盒子时，又恢复到盒子默认设置的样式；而单击盒子时则在盒子内输出动态信息，显示当前盒子的宽度和高度。代码如下，演示效果如图 14.6 所示。

```
<script>
window.onload = function(){
    var box = document.getElementById("box");                    //获取盒子的引用
    box.onmouseover = function(){                                //定义鼠标指针经过时的事件处理函数
        box.style.setProperty("background-color", "blue", "");  //设置背景色为蓝色
        box.style.setProperty("border", "solid 50px red", "");   //设置边框为 50 像素的红色实线
    }
    box.onclick = function(){                                    //定义鼠标单击时的事件处理函数
        box .innerHTML = (box.style.item(0) + ":" + box.style.getPropertyValue("width"));
                                                                 //显示盒子的宽度
        box .innerHTML = box .innerHTML + "<br>" +  (box.style.item(1) + ":" + box.style.getPropertyValue
("height"));                                                     //显示盒子的高度
    }
    box.onmouseout = function(){                                 //定义鼠标指针移出时的事件处理函数
        box.style.setProperty("background-color", "red", "");    //设置背景色为红色
        box.style.setProperty("border", "solid 50px blue", "");  //设置边框为 50 像素的蓝色实线
    }
}
</script>

<div id="box" style="width:100px; height:100px; background-color:red; border:solid 50px blue;"></div>
```



默认显示效果　　鼠标指针经过效果　　鼠标单击效果

图 14.6　设计动态交互样式效果





【示例 5】针对示例 4，本示例使用一种快捷方式设计相同的交互效果，这样能够兼容 IE 早期版本，页面代码如下。

```
<script>
window.onload = function(){
    var box = document.getElementById("box");              //获取盒子的引用
    box.onmouseover = function(){
        box.style.backgroundColor = "blue";                //设置背景样式
        box.style.border = "solid 50px red";               //设置边框样式
    }
    box.onclick = function(){//读取并输出行内样式
        box .innerHTML = "width:" + box.style.width;
        box .innerHTML = box .innerHTML + "<br>" +  "height:" + box.style.height;
    }
    box.onmouseout = function(){//设计鼠标指针移出之后，恢复默认样式
        box.style.backgroundColor = "red";
        box.style.border = "solid 50px blue";
    }
}
</script>

<div id="box" style="width:100px; height:100px; background-color:red; border:solid 50px blue;"></div>
```

【拓展】

非 IE 浏览器也支持 style 快捷访问方式，但是它无法获取 style 对象中指定序号位置的属性名称，此时可以使用 cssText 属性读取全部 style 属性值，再借助 JavaScript 方法把返回字符串劈开为数组。

【示例 6】在本示例中，使用 cssText 读取全部行内样式字符串，然后使用 String 的 split()方法把字符串劈开为数组，使用 for in 语句遍历数组，逐一读取每个样式，再使用 split()方法劈开属性和属性名，最后格式化输出显示。代码如下，演示效果如图 14.7 所示。

```
<script>
window.onload = function(){
    var box = document.getElementById("box");              //获取盒子的引用
    var str = box.style.cssText;                           //读取盒子全部行内样式
    var a = str.split(";");                                //把行内样式字符串转换为数组
    var temp="";
    for(var b in a){                                       //遍历行内样式
        var prop = a[b].split(":");                        //把每个样式字符串劈开为数组
        if(prop[0])                                        //如果存在属性，则输出显示
```





```
                temp += b + " ： " + prop[0] + " = " + prop[1] + "<br>";
        }
        box.innerHTML = "box.style.cssText = " + str;
        box.innerHTML = box.innerHTML + "<br><br>" + temp;        //把格式化后的行内样式输出显示
    }
</script>

<div id="box" style="width:600px; height:200px; background-color:#81F9A5; border:solid 2px blue;padding:
10px"></div>
```

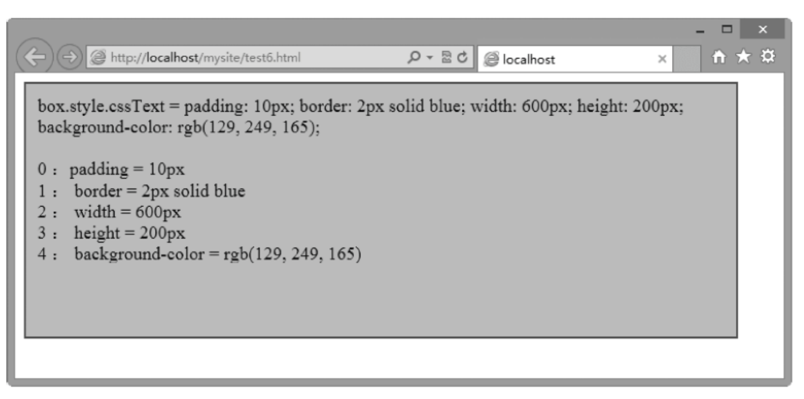


图 14.7　使用 cssText 属性获取行内样式

使用 getAttribute()方法也可以获取 style 属性值。不过该方法返回值保留 style 属性值的原始模样，而 cssText 属性返回值可能经过浏览器处理，且不同浏览器返回值格式略有不同。

【示例 7】本示例将修改示例 6 的代码，使用 getAttribute()方法获取行内样式字符串信息，代码如下。

```
<script>
window.onload = function(){
    var box = document.getElementById("box");
    var str = box.getAttribute("style");
    var a = str.split(";");
    var temp="";
    for(var b in a){
        var prop = a[b].split(":");
    if(prop[0])
        temp += b + " ： " + prop[0] + " = " + prop[1] + "<br>";
    }
    box.innerHTML = "box.style.cssText = " + str;
    box.innerHTML = box.innerHTML + "<br><br>" + temp;
```




```
}
</script>

<div id="box" style="width:600px; height:200px; background-color:#81F9A5; border:solid 2px blue;padding:
10px"></div>
```



Note

视频讲解

14.4.2 使用 styleSheets 对象

document 对象包含一个 styleSheets 属性集合，它保存了文档中所有的样式表，包括内部样式表和外部样式表。

styleSheets 为每个样式表定义了一个 cssRules 对象，用来包含指定样式表中所有的规则（样式）。但是 IE 不支持 cssRules 对象，而预定义了 rules 对象表示样式表中的规则。

为了兼容主流浏览器，在使用前应该检测用户所使用浏览器的类型，以便调用不同的对象，代码如下。

```
var cssRules = document.styleSheets[0].cssRules || document.styleSheets[0].rules;
```

在上面的代码中，先判断浏览器是否支持 cssRules 对象，如果支持则使用 cssRules（非 IE 浏览器），否则使用 rules（IE 浏览器）。

【示例】 在本示例中，通过<style>标签定义一个内部样式表，为页面中的<div id="box">标签定义 4 个属性：宽度、高度、背景色和边框。然后在脚本中使用 styleSheets 访问这个内部样式表，把样式表中第一个样式的所有规则读取出来，在盒子中输出显示。代码如下，效果如图 14.8 所示。

```
<style type="text/css">
#box {
    width: 400px;
    height: 200px;
    background-color:#BFFB8F;
    border: solid 1px blue;
}
</style>
<script>
window.onload = function(){
    var box = document.getElementById("box");
    var cssRules = document.styleSheets[0].cssRules || document.styleSheets[0].rules;//判断浏览器类型
    box.innerHTML = "<h3>盒子样式</h3>"
    box.innerHTML += "<br>边框：" + cssRules[0].style.border; //读取 cssRules 的 border 属性
    box.innerHTML += "<br>背景："+ cssRules[0].style.backgroundColor;//读取 cssRules 的 background-color 属性
    box.innerHTML += "<br>高度：" + cssRules[0].style.height;//读取 cssRules 的 height 属性
    box.innerHTML += "<br>宽度：" + cssRules[0].style.width;//读取 cssRules 的 width 属性
```





```
}
</script>

<div id="box"></div>
```



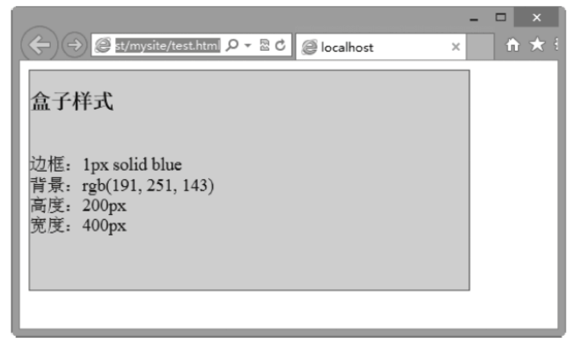


图 14.8　使用 styleSheets 访问内部样式表

提示：cssRules（或 rules）的 style 对象在访问 CSS 属性时，使用的是 CSS 脚本属性名，因此所有属性名称中不能使用连字符。例如：

```
cssRules[0].style.backgroundColor;
```

这与行内样式中的 style 对象的 setProperty()方法不同，setProperty()方法使用的是 CSS 属性名。例如：

```
box.style.setProperty("background-color", "blue", "");
```

14.4.3　访问样式



styleSheets 包含文档中所有的样式表，用户可以通过下标访问每个样式表，每个数组元素代表一个样式表，数组的索引位置是根据样式表在文档中的位置决定的。每个<style>标签包含的所有样式表示一个内部样式表，每个独立的 CSS 文件表示一个外部样式表。

【示例】本示例演示了如何准确找到指定样式表中的样式属性。

第 1 步，启动 Dreamweaver，新建 CSS 文件，保存为 style1.css，存放在根目录下。

第 2 步，在 style1.css 中输入下面的样式代码，定义一个外部样式表。

```
@charset "utf-8";
body { color:black; }
p { color:gray; }
div { color:white; }
```

第 3 步，新建 HTML 文档，保存为 test.html，保存在根目录下。

第 4 步，使用<style>标签定义一个内部样式表，设计如下样式。




Note

```
<style type="text/css">
#box { color:green; }
.red { color:red; }
.blue { color:blue; }
</style>
```

第 5 步，使用<link>标签导入外部样式表文件 style1.css，代码如下。

```
<link href="style1.css" rel="stylesheet" type="text/css" media="all" />
```

第 6 步，在文档中插入一个<div id="box">标签，代码如下。

```
<div id="box"></div>
```

第 7 步，使用<script>标签在头部位置插入一段脚本。设计在页面初始化完毕后，使用 styleSheets 访问文档中第 2 个样式表，然后再访问该样式表的第 1 个样式中的 color 属性，代码如下。

```
<script>
window.onload = function(){
    var cssRules = document.styleSheets[1].cssRules || document.styleSheets[1].rules;
    var box = document.getElementById("box");
    box.innerHTML =  "第二个样式表中第一个样式的 color 属性值 = " + cssRules[0].style.color;
}
</script>
```

第 8 步，保存页面，整个文档的代码如下。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<style type="text/css">
#box { color:green; }
.red { color:red; }
.blue { color:blue; }
</style>
<link href="style1.css" rel="stylesheet" type="text/css" media="all" />
<script>
window.onload = function(){
    var cssRules = document.styleSheets[1].cssRules || document.styleSheets[1].rules;
    var box = document.getElementById("box");
    box.innerHTML =  "第二个样式表中第一个样式的 color 属性值 = " + cssRules[0].style.color;
```





```
}
</script>
</head>
<body>
<div id="box"></div>
</body>
</html>
```

最后，在浏览器中预览页面，则可以看到访问的 color 属性值为 black，如图 14.9 所示。

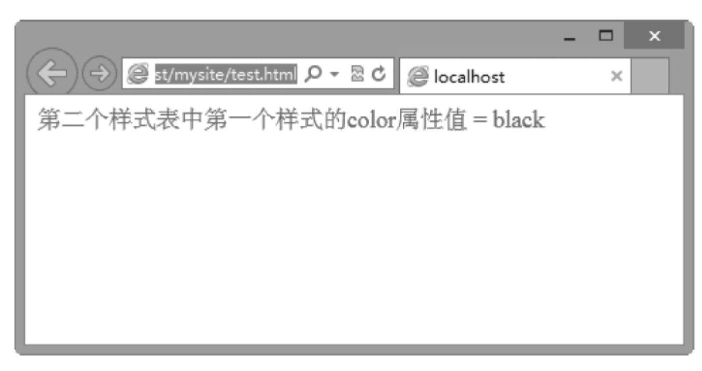


图 14.9　使用 styleSheets 访问外部样式表

提示：在上面的示例中，styleSheets[1]表示外部样式表文件（style1.css），而 cssRules[0]表示外部样式表文件中的第一个样式。cssRules[0].style.color 可以获取外部样式表文件中第一个样式中 color 属性的声明值。反之，如果把<link>标签放置在内部样式表的上面，即代码如下。

```
<head>
<link href="style1.css" rel="stylesheet" type="text/css" media="all" />
<style type="text/css">
#box { color:green; }
.red { color:red; }
.blue { color:blue; }
</style>
</head>
```

上面脚本将返回内部样式表中第一个样式中的 color 属性声明值，即为 green。如果把外部样式表转换为内部样式表，或者把内部样式表转换为外部样式表文件，不会影响 styleSheets 的访问。因此，样式表和样式的索引位置是不受样式表类型和样式的选择符限制的。任何类型的样式表（不管是内部的，还是外部的）都在同一个平台上按在文档中解析位置进行索引。同理，不同类型选择符的样式在同一个样式表中也是根据先后位置进行索引。



14.4.4 编辑样式



Note



视频讲解

cssRules 的 style 对象不仅可以访问属性，还可以设置属性值。

【示例】在本示例中，样式表中包含 3 个样式，其中蓝色样式类（.blue）定义字体显示为蓝色。利用脚本修改该样式类（.blue 规则）字体颜色为浅灰色（#999）。代码如下，最后显示效果如图 14.10 所示。

```
<style type="text/css">
#box { color:green; }
.red { color:red; }
.blue { color:blue; }
</style>
<script>
window.onload = function(){
    var cssRules = document.styleSheets[0].cssRules || document.styleSheets[0].rules;
    cssRules[2].style.color="#999";          //修改样式表中指定属性的值
}
</script>

<p class="blue">原为蓝色字体，现在显示为浅灰色。</p>
```



图 14.10　修改样式表中的样式

提示：上述方法修改样式表中的类样式，会影响其他对象或其他文档对当前样式表的引用，因此在使用时请务必谨慎。

14.5　案 例 实 战

本节将通过多个案例帮助读者上机练习使用 JavaScript 控制 CSS 的方法。



视频讲解

14.5.1 设计显示和隐藏

简单的隐藏元素可以通过 style.display 属性来实现。



【示例 1】 本示例能够遍历结构中所有的 p 元素，并把 class 属性值不为 main 的段落文本全部隐藏，代码如下。

```
<p>p1</p>
<p class="main">p2</p>
<p>p3</p>
<script>
var p = document.getElementsByTagName("p");
for(var i = 0; i < p.length; i ++ ){
    if(p[i].className == "main") continue;
    // 如果 class 属性值为 main，则跳过
    p[i].style.display = "none";              // 隐藏元素
}
</script>
```

恢复 style.display 属性的默认值，只需设置 style.display 属性值为空字符串（style.display = ""）。

【示例 2】 由于显示和隐藏是交互设计中经常用到的技巧，所以有必要对其进行功能封装，以实现代码重用和灵活应用，并能够兼容不同浏览器。

当指定元素和布尔值参数时，元素能够根据布尔值 true 或 false 决定是否进行显示或隐藏，如果不指定第 2 个布尔值参数，则函数将对元素进行显示或隐藏切换，代码如下。

```
// 设置或切换元素的显示或隐藏
// 参数：e 表示要显示或隐藏的元素，b 是一个布尔值，当为 true 时，将显示元素 e；
当为 false 时，将隐藏元素 e。如果省略参数 b，则根据元素 e 的显示状态，进行显示或
隐藏切换
// 返回值：无
function display(e, b){
    // 监测第二个参数的类型。如果该参数存在且不为布尔值，则抛出异常
    if(b && (typeof b != "boolean")) throw new Error("第二个参数应该是布尔值!");
    var c = getStyle(e, "display");            // 获取当前元素的显示属性值
    (c != "none") && (e._display = c);         // 记录元素的显示性质，并存储到元素的属性中
    e._display = e._display || "";             // 如果没有定义显示性质，则赋值为空字符串
    if(b || (c == "none") ){                   // 第二个参数值为 true 或者元素隐藏
        e.style.display = e._display;          // 调用元素的_display 属性值恢复元素或显示元素
    }
    else{
        e.style.display = "none";              // 隐藏元素
    }
}
```




下面在页面中设置一个向右浮动的元素 p。连续调用 3 次 display()函数后，则相当于隐藏元素，代码如下。



```
<p style="float:right; border:solid 1px red; width:100px;
height:100px;">p1</p>
<script>
var p = document.getElementsByTagName("p")[0];
display(p);                    // 切换隐藏
display(p);                    // 切换显示
display(p);                    // 切换隐藏
</script>
```

不管元素是显示还是隐藏，如果按如下方式调用，则会显示出来，元素显示为原来的状态，代码如下。

```
display(p , true);                    // 强制显示
```



14.5.2 设计不透明度

所有现代浏览器都支持元素的透明度，但是不同浏览器对于元素透明度的设置方法不同。IE 浏览器支持 filters 滤镜集，而支持 DOM 标准的浏览器认可 style.opacity 属性。同时，它们设置值的范围也不同，IE 的 opacity 属性值范围为 0～100，其中 0 表示完全透明，而 100 表示不透明。而支持 style.opacity 属性浏览器的设置值范围是 0～1，其中 0 表示完全透明，而 1 表示不透明。

【示例 1】 为了兼容不同的浏览器，可以把设置元素透明度的功能进行函数封装，代码如下。

```
// 设置元素的透明度
// 参数：e 表示要预设置的元素，n 表示一个数值，取值范围为 0～100，如果省略，
则默认为 100，即不透明显示元素
// 返回值：无
function setOpacity(e, n){
    var n = parseFloat(n);                    // 把第二个参数转换为浮点数
    if(n && (n>100) || !n) n=100;
    // 如果第二个参数存在且值大于 100，或者不存在该参数，则设置其为 100
    if(n && (n<0))    n =0;                   // 如果第二个参数存在且值小于 0，则设置其为 0
    if (e.filters){                           // 兼容 IE 浏览器
        e.style.filter = "alpha(opacity=" + n + ")";
    }
    else{                                     // 兼容 DOM 标准
        e.style.opacity = n / 100;
    }
}
```



在获取元素的透明度时，应注意在 IE 浏览器中不能够直接通过属性读取，而应借助 filters 集合的 item()方法获取 Alpha 对象，然后读取它的 opacity 属性值。

【示例 2】为了避免在读取 IE 浏览器中元素的透明度时发生错误，建议使用 try 语句包含读取语句，代码如下。

```
// 获取元素的透明度
// 参数：e 表示要预设置的元素
// 返回值：元素的透明度值，范围为 1～100
function getOpacity(e){
    var r;
    if ( ! e.filters){
        if (e.style.opacity) return parseFloat(e.style.opacity) * 100;
    }
    try{
        return e.filters.item('alpha').opacity
    }
    catch(o){
        return 100;
    }
}
```

14.5.3　设计运动对象



运动效果主要通过动态修改元素的坐标来实现。设计的关键有以下两点。

☑　应考虑元素的初始化坐标、最终坐标，以及移动坐标等定位要素。如果参照物相同，则这个问题比较好解决。

☑　应考虑移动的速度、频率等问题。移动可以借助定时器来实现，但效果的模拟涉及算法问题，不同的算法可能会设计出不同的移动效果，如匀速运动、加速运动和减速运动。在 Flash 动画设计中，就专门提供了一个 Tween 类，利用它可以模拟出很多运动效果，如缓动、弹簧震动等，其技术核心是算法设计问题。算法看似很高深，其实通俗一点讲，就是通过数学函数计算定时器每次触发时移动的距离。

【示例】本示例演示如何设计一个简单的元素滑动效果。通过指定元素和移动的位置，以及移动的步数，可以设计按一定的速度把元素从当前位置移动到指定的位置。本示例引用前面介绍的 getB()方法，该方法能够获取当前元素的绝对定位坐标值，代码如下。

```
// 简单的滑动函数
// 参数：e 表示元素，x 和 y 表示要移动到的最后坐标位置（相对包含块），t 表示元素移动的步数
function slide(e, x, y, t){
    var t = t || 100;                    // 初始化步数，步数越大，速度越慢，移动的过程越逼真，但
```






```
                                          是中间移动的误差就越明显
    var o = getB(e);                      // 当前元素的绝对定位坐标值
    var x0 = o.x;
    var y0 = o.y;
    var stepx = Math.round((x - x0) / t);
    // 计算 x 轴每次移动的步长，由于像素点不可用小数，所以会存在一定的误差
    var stepy = Math.round((y - y0) / t);   // 计算 y 轴每次移动的步长
    var out =   setInterval(function(){     // 设计定时器
    var o = getB(e);                      // 获取每次移动后的绝对定位坐标值
    var x0 = o.x;
    var y0 = o.y;
    e.style["left"] = (x0 + stepx) + 'px';    // 定位每次移动的位置
    e.style["top"] = (y0 + stepy) + 'px';     // 定位每次移动的位置
    if (Math.abs(x - x0) <= Math.abs(stepx) || Math.abs(y - y0) <=
Math.abs(stepy)) {                        // 如果到终点坐标的距离小于步长，则停止循环执行，并校正
                                          元素的最终坐标位置
        e.style["left"] = x + 'px';
        e.style["top"] = y + 'px';
        clearTimeout(out);
    };
    }, 2)
};
```

使用时应该定义元素绝对定位或相对定位显示状态，否则移动无效。在网页动画设计中，一般都使用这种定位移动的方式来实现，代码如下。

```
<style type="text/css">
.block {width:20px; height:20px; position:absolute; left:200px;
top:200px; background-color:red; }
</style>
<div class="block" id="block1"></div>
<script>
temp1 = document.getElementById('block1');
slide(temp1, 400, 400,60);
</script>
```



14.5.4 设计渐变效果

渐隐渐显效果主要通过动态修改元素的透明度来实现。





【示例】本示例演示如何实现简单的渐隐渐显动画效果，主要涉及 setOpacity()函数的调用，代码如下。

```
// 渐隐渐显动画显示函数
// 参数：e 表示渐隐渐显元素，t 表示渐隐渐显的速度，值越大渐隐渐显速度越慢，
io 表示渐隐或渐显的方式，取值 true 表示渐显，取值 false 表示渐隐
function fade(e, t, io){
    var t = t || 10;                          // 初始化渐隐渐显速度
    if(io){                                   // 初始化渐隐渐显方式
    var i = 0;
    }else{
    var i = 100;
    }
    var out = setInterval(function(){         // 设计定时器
    setOpacity(e, i);                         // 调用 setOpacity()函数
    if(io) {                                  // 根据渐隐或渐显方式决定执行效果
        i ++ ;
        if(i >= 100)   clearTimeout(out);
    }
    else{
        i-- ;
        if(i <= 0)   clearTimeout(out);
    }
    }, t);
}
```

下面调用该函数，代码如下。

```
<style type="text/css">
.block {width:200px; height:200px; background-color:red; }
</style>
<div class="block" id="block1"></div>
<script>
e = document.getElementById('block1');
fade(e,50,true);                  // 应用渐隐渐显动画效果
</script>
```

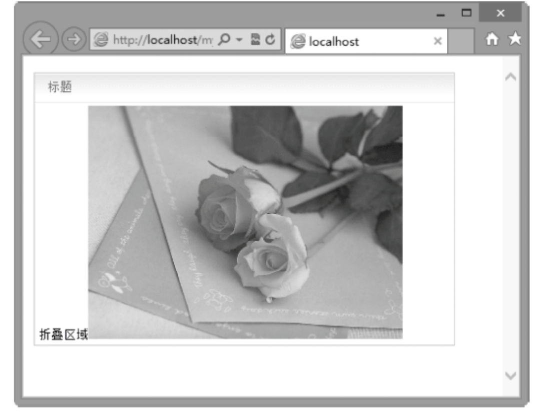
14.5.5　设计折叠面板



折叠面板在网页中的应用范围比较广，从技术角度分析，它主要利用 CSS 隐藏和显示属性，借助 JavaScript 脚本进行动态控制。本节设计的折叠面板演示效果如图 14.11 和图 14.12 所示。







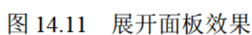


图 14.11　展开面板效果

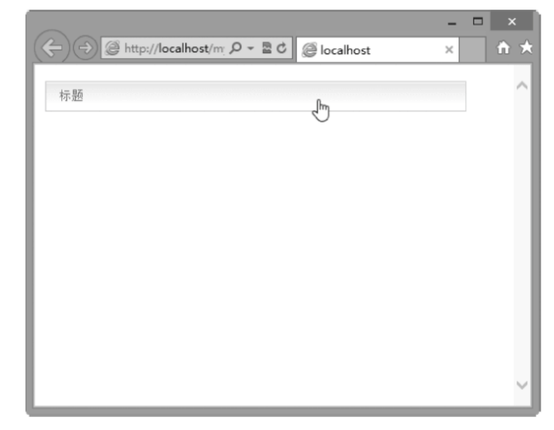


图 14.12　折起面板效果

【操作步骤】

第 1 步，启动 Dreamweaver，新建文档，保存为 test.html。构建 HTML 结构。结构没有特殊要求，使用任何两个标签均可实现折叠效果，但是从语义角度来考虑，使用定义列表是最佳语义结构选择。dl 元素负责构建折叠面板的外框，dt 元素负责构建折叠面板的标题栏，而 dd 元素负责构建面板主体包含框，代码如下。

```
<dl>
    <dt>标题</dt>
    <dd>折叠区域<img src="images/3.jpg" width="300" /></dd>
</dl>
```

第 2 步，在设计折叠行为之前，我们先假设这是一个普通的列表结构。然后使用 CSS 来控制定义列表框的表现效果，代码如下。

```
<style type="text/css">
dl {/* 定义列表框样式 */
    width:400px;                          /* 定义折叠面板的宽度，可自定义 */
    border:solid 1px #ccc;                /* 边框，可自定义*/
    font-size:12px;                       /* 字体大小，可自定义*/
    }
dt {/* 列表标题样式 */
    background:#7FECAD url(images/green.gif) repeat-x; /* 用背景图定义渐变标题背景 */
    color:#71790C;                        /* 字体颜色，可自定义*/
    height:28px;                          /* 标题高度 */
    line-height:28px;                     /* 行高，间接实现垂直对齐 */
    padding-left:1em;                     /* 增加左侧空隙 */
    border-bottom:solid 1px #efefef;      /* 底部边框样式 */
```



```
        cursor:pointer;                      /* 鼠标指针为手形 */
        }
    dd {/* 列表项样式 */
        padding:2px 4px;                     /* 增加内容框内边距 */
        margin:0;                            /* 清除缩进 */
        }
    </style>
```

第 3 步，在上面样式表的基础上定义两个类样式，分别用来隐藏和显示对象，代码如下。

```
    .expand { overflow:visible; }            /* 展开面板时，显示所有内容区域 */
    .collapse {                              /* 折叠面板时，仅显示标题区域 */
        height:28px;                         /* 限制列表包含框高度，使其等于标题栏高度 */
        overflow:hidden;                     /* 强制隐藏多出的区域 */
            }
```

第 4 步，完成结构层和表现层的设计，下面就来设计交互层。在 JavaScript 脚本中定义一个函数 Switch()用作展开和折叠的开关，代码如下。

```
    <script>
    function Switch(dt){ //折叠控制函数
        var dl = dt.parentNode;                                   //获取标题栏的父包含框
        if(dl.className == "collapse")dl.className = "expand";   //如果为折叠，则展开类样式
        else dl.className = "collapse";                           //相反调用折叠类样式
    }
    </script>
```

第 5 步，完成脚本函数的设计，然后把它绑定到标题栏的 onclick 事件属性上面，代码如下。

```
    <dt onclick="Switch(this)">标题</dt>
```

这里使用 this 关键字作为参数进行传递，它代表当前 dt 元素的引用。至此，整个折叠面板的设计就完成了。

14.5.6　设计工具提示



视 频 讲

Tooltip（工具提示）是一种比较实用的 JavaScript 应用。当为一个元素（一般为超链接 a 元素）定义 title 属性时，会在鼠标指针经过时显示提示信息，这些提示能够详细描绘经过对象的包含信息，这对于超链接（特别是图像式超链接）非常有用。同时，搜索引擎也喜欢检索这些信息。

设计思路：使用 DOM 技术获取 title（或其他属性）中的提示信息，然后把这些属性删除，再利用 JavaScript 脚本动态生成一个浮动的层，在层中显示这些提示信息，最后利用 Even 事件对象的鼠标指针坐标属性进行定位。如果结合 CSS 技术，可以把这些浮动的层设计成不同样式，以此达到个





Note

性化设计的要求。

【示例 1】本示例不涉及结构层和表现层的设计，为了化繁为简，这里先就一个简单的案例来探索 Tooltip 脚本的实现过程。

第 1 步，启动 Dreamweaver，新建文档，保存为 test.html。在文档中设计如下超链接，其提示信息设置为“title="提示信息"”。下面尝试把这个提示信息提取出来，然后删除该属性，最后使用一个新创建的 div 元素动态显示它的位置，并借助 CSS 美化一下该 div 元素，代码如下。

```
<a href="#" title="提示信息" target="_blank">超链接文本</a>
```

第 2 步，在脚本中获取超链接元素 a 以及该标签设置的 title 属性值，代码如下。

```
var a = document.getElementsByTagName("a")[0];          //获取 a 元素的引用
var tit   = a.getAttribute("title");                     //获取 title 属性值，并存储到一个变量中
```

第 3 步，获取 title 属性值之后，删除该属性，避免它干扰设计，代码如下。

```
a.removeAttribute("title");                              //移出 title 属性
```

第 4 步，创建一个 div 元素和一个文本节点，把 title 属性值赋予文本节点，然后把文本节点增加到 div 元素内，设置 div 元素样式为绝对定位，并增加一个 class 属性和值，以便于在 CSS 样式表中对该 div 元素进行更个性化地控制，代码如下。

```
var div = document.createElement("div");                 //创建 div 元素节点
vartxt = document.createTextNode(tit);                   //创建文本节点，并显示 title 属性值
div.style.position = "absolute";                         //为 div 元素定义一个绝对定位
div.setAttribute("class", "title");                      //为 div 元素增加一个类样式，兼容非 IE
div.setAttribute("className", "title");                  //为 div 元素增加一个类样式，兼容 IE
div.appendChild(txt);                                    //获取 title 属性值，并传递给 div 元素
```

第 5 步，为超链接 a 元素绑定鼠标指针经过和鼠标指针移出的事件处理函数。设计当鼠标指针移过超链接文本时，把创建的 div 元素节点增加到该 a 元素中，而当鼠标指针移出超链接文本时，把 a 元素中增加的 div 节点删除，代码如下。

```
a.onmouseover = function(){                                   //鼠标指针经过事件处理函数
    a.appendChild(div);                                       //把 div 元素增加到 a 元素中
}
a.onmouseout = function(){                                    //鼠标指针移出事件处理函数
    a.removeChild(div);                                       //把 a 元素中的 div 元素删除
}
```

第 6 步，定义鼠标指针移动事件处理函数，实时跟踪鼠标指针的坐标，并利用该坐标来定位创建的 div 元素的显示位置，以实现它始终显示在鼠标指针的右下角。





第 7 步，在 CSS 样式表中为 title 类定义类样式，代码如下。

```
<style type="text/css">
.title {/* 提示框类样式 */
        padding:4px 8px;              /* 增加内侧补白 */
    border:solid 2px red;             /* 设计边框样式 */
    background:blue;                  /* 定义背景为蓝色 */
    color:#fff;                       /* 定义字体为白色 */
    text-decoration:none;             /* 清除受 a 元素影响而产生的下划线 */
}
</style>
```

至此，整个提示框效果就设计完成了。在浏览器中预览，则显示效果如图 14.13 和图 14.14 所示。



图 14.13　指定元素的提示框演示效果（1）



图 14.14　指定元素的提示框演示效果（2）

【示例 2】示例 1 仅就页面中某个具体的超链接来定义提示框，但是在实际设计中是无法预测到页面中到底有多少超链接，为此需要使用遍历 a 元素节点集合技术来实现动态为页面中所有超链接设计提示框，演示效果如图 14.15 和图 14.16 所示。



图 14.15　为页面中所有超链接设计提示框演示效果（1）



图 14.16　为页面中所有超链接设计提示框演示效果（2）

实现上述演示效果，本示例依然采用上节示例的设计思路，只不过中间增加了 for 循环结构来遍历文档中所有的超链接元素 a。

第 1 步，复制 test.html，另存为 test2.html。在页面初始化事件中绑定一个事件处理函数。然后获取页面内所有 a 元素的引用，代码如下。




```
window.onload = function(){
    var a = document.getElementsByTagName("a");          //创建 div 元素节点
}
```

第 2 步，在该事件处理函数内，完成上节示例的操作，代码如下。

```
for(var i = 0; i < a.length; i ++ ){                      //遍历页面内所有 a 元素
    tit = a[i].getAttribute("title");                     //获取 a 元素的 title 属性值
    if(tit)   a[i].removeAttribute("title");              //如果属性值存在，则删除该属性
    var div = document.createElement("div");              //创建 div 元素节点
    var txt = document.createTextNode(tit);               //创建并把提示信息赋予文本节点
    div.setAttribute("class", "title");                   //为 div 元素增加类属性，兼容 FF
    div.setAttribute("className", "title");               //为 div 元素增加类属性，兼容 IE
    div.style.position = "absolute";                      //绝对定位 div 元素
    div.appendChild(txt);                                 //把文本节点增加到 div 元素
}
```

第 3 步，设计鼠标指针经过和移出的事件处理函数，以实现增加和删除 div 到 a 元素。考虑到在函数体内定义闭包是无法与外界进行数据交流的，为此主动为闭包函数传递外部动态参数，代码如下。

```
a[i].onmouseover = (function(i,div){              //鼠标指针经过时的事件处理函数
    return function(){                            //返回处理函数
        a[i].appendChild(div);                    //为 a 元素增加 div 元素
    }
})(i,div);                                  //为闭包函数传递参数，i 表示循环变量值，div 表示引用
a[i].onmouseout = (function(i,div){         //鼠标指针移出时的事件处理函数
    return function(){                      //返回处理函数
        a[i].removeChild(div);              //为 a 元素移出 div 元素
    }
})(i,div);
```

第 4 步，设计鼠标指针移动的事件处理函数。为了能够兼容不同主流浏览器，以及考虑浏览器窗口可能会出现滚动条，这里使用多个条件结构进行判断来设置指针的坐标值，代码如下，读者需要了解 Event 对象的属性，详细内容可以参阅上一章讲解。

```
a[i].onmousemove = (function(div,e){//第 1 个参数表示定位元素，第 2 个参数表示事件参数
    return function(e) {//闭包内返回函数体
        var posx = 0, posy = 0;                   //定义两个局部变量，用来存储鼠标指针的坐标
        //判断当前浏览器，如果为 IE，则使用 window.event 获取鼠标指针
        if(e == null) e = window.event;
        //判断是否支持 pageX 或 pageY 事件属性，如果支持表示浏览器支持 DOM 2.0（如 FF 等），此时
```



```
            可以使用这两个属性获取鼠标指针在窗口中的坐标
            if(e.pageX || e.pageY){
                posx = e.pageX;
                posy = e.pageY;
            }
            else if(e.clientX || e.clientY){//如果不支持 pageX 或 pageY，则使用 clientX 或 clientY
                //如果支持 document.documentElement.scrollTop 属性，则计算指针坐标
                if(document.documentElement.scrollTop){
                    posx = e.clientX + document.documentElement.scrollLeft;
                    posy = e.clientY + document.documentElement.scrollTop;
                }else{//否则使用传统的方法来计算指针的坐标位置
                    posx = e.clientX + document.body.scrollLeft;
                    posy = e.clientY + document.body.scrollTop;
                }
            }
            div.style.top = (posy + 20) + "px";          //把鼠标指针的 y 坐标作为定位值赋予 div
            div.style.left = (posx + 10) + "px";         //把鼠标指针的 x 坐标作为定位值赋予 div
        }
    })(div);
```

提示： documentElement 在 DOM 2.0 中表示 html 元素，因此要获取当前页面的滚动条纵坐标位置，可以使用如下方法。

```
document.documentElement.scrollTop;
```

如果浏览器不支持 documentElement 对象，则可以使用如下方法获取滚动条纵坐标位置。

```
document.body.scrollTop;
```

这里的 body 对象表示 body 元素，而 document 表示文档对象，scrollTop 属性表示滚动条的纵坐标。在标准 W3C 下，document.body.scrollTop 始终为 0，需要使用 document.documentElement.scrollTop 来获取滚动条坐标。IE 浏览器从 5.5 版本开始不再支持 document.body.scrollTop 和 document.body.scrollLeft 方法，所以在编程中一般使用上面方法来进行判断。

14.6　在线练习

CSS 脚本样式练习。感兴趣的读者可以扫码练习。





团购类型网站的布局与设计

团购网就是团体购买的网络组织平台，即互不认识的消费者，借助互联网的“网聚人的力量”来聚集资金，加大与商家的谈判能力，以求用最优的价格购买商品和服务。根据薄利多销、量大价优的原理，商家可以给出低于零售价格的团购折扣和单独购买得不到的优质服务。在2009年的网交会上，马云提出了“网货2.0”的概念，即消费者按需定制、厂商柔性生产的C2B消费模式，如今团购网由购买者集体提出条件与商家谈判，由商家根据购买者的要求进行生产或者定价的运营模式，正是体现出了网货2.0的概念，也是未来以顾客为中心进行网络营销的良好开端。

由于团购市场的兴起和逐渐成熟，优胜劣汰，自然而然就会产生一批成功的网站，比较出名的有窝窝团（http://www.55tuan.com/，如图15.1所示）、拉手网（http://www.lashou.com，如图15.2所示）、24券（http://www.24quan.com/）、糯米网（http://www.nuomi.com）、去哪儿团购（http://tuan.qunar.com/）和满座网（http://www.manzuo.com/）等。



图15.1　窝窝团　　　　图15.2　拉手网





15.1　产 品 策 划

团购类型的网站结构很简单，就是以每一期的团购内容为单元，内容包括团购的标题、大图片展示、价格显示和相关操作。所以也就不用考虑太多页面的逻辑结构问题，只要把每期的团购显示在页面上即可。一般来说，团购类型的网站多采用 3 列多行的方式布局，如图 15.3 所示，也有两列多行的方式，如图 15.4 所示。

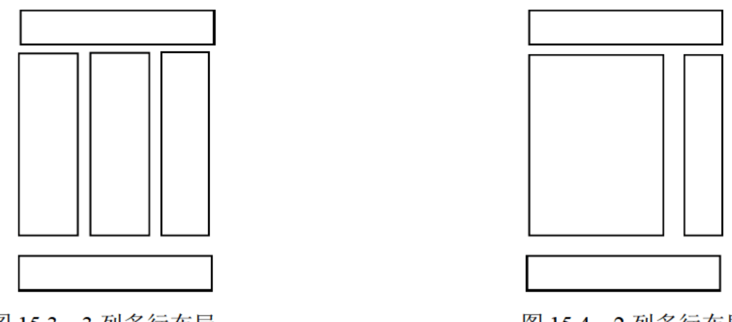


图 15.3　3 列多行布局　　　　图 15.4　2 列多行布局

与其他团购网站一样，美团网具备典型的团购网特点。除了页眉和页脚，中间主题内容基本上也是分为 3 栏。其中左边和中间的一栏为具体团购的每一个单元，右边的一栏为团购相关单元。而每一个团购单元里面标题文字都会设置成比较大的字号，而完全不管页面是否美观。最重要的价格显示区都会设计得与众不同且比较明显。

就结构布局来说，团购网这一类型的网站是做得非常到位的，即使页面可以拉得很长很长，但我们不用拉滚动条也基本上能猜到下面的页面是什么内容，无非就是以前的某期团购内容，越往下面的页面，时间也越靠后。

说到美团网的配色设计方面，也是比较出色的。首先，一般团购网都会选择暖色调作为主色，但美团网却选了蓝色和绿色，还配上页眉页脚的深灰色。再者，美团网的细节也是做得非常到位的。我们看看网站的 logo 文字“美团”就可以感受到，用灰色和白色就可以做出好像水晶一样的效果，很精致耐看。还有网站的背景也利用起来，用了一张背景图而不是用一个纯色。

其实设计往往是赢在细节上，做一个细节，可能对整体没什么影响，做两个细节也还不能产生什么影响，但当所有细节合在一起整体看时，效果就出来了。

15.2　画板和设计

根据策划的基本思路，在做产品设计时，美团网在准确表达团购内容的布局基础上适当设计一些个性的表达元素，整个网站包括网站 logo、导航栏、每期的团购相关信息、团购周边相关、问题反馈、版权信息等版块内容，效果如图 15.5 所示。


Note

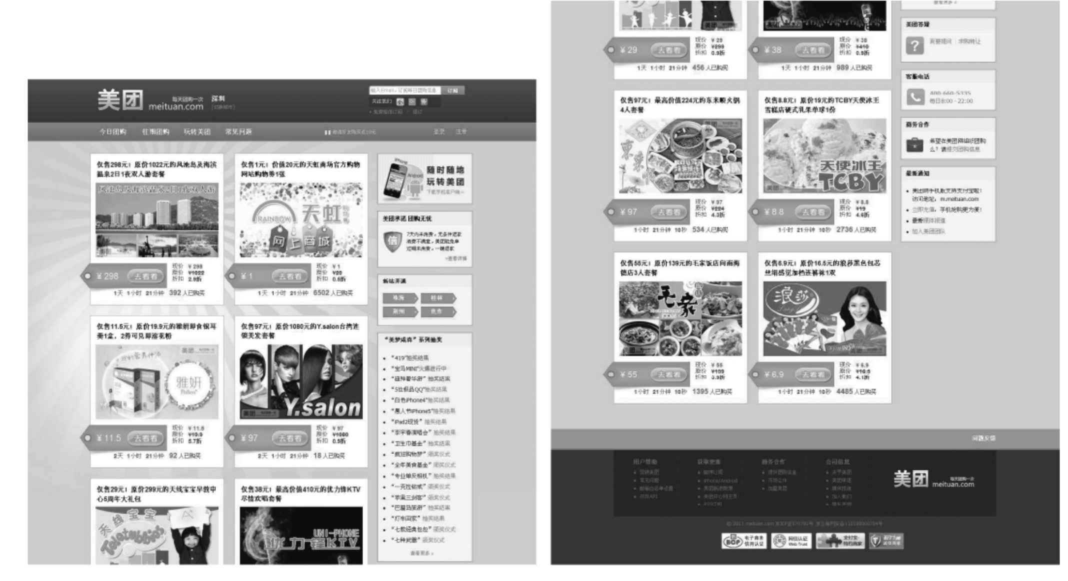


图 15.5　网站效果图

根据这些模块，初步在稿纸上画出页面布局的草图，如图 15.6 所示。

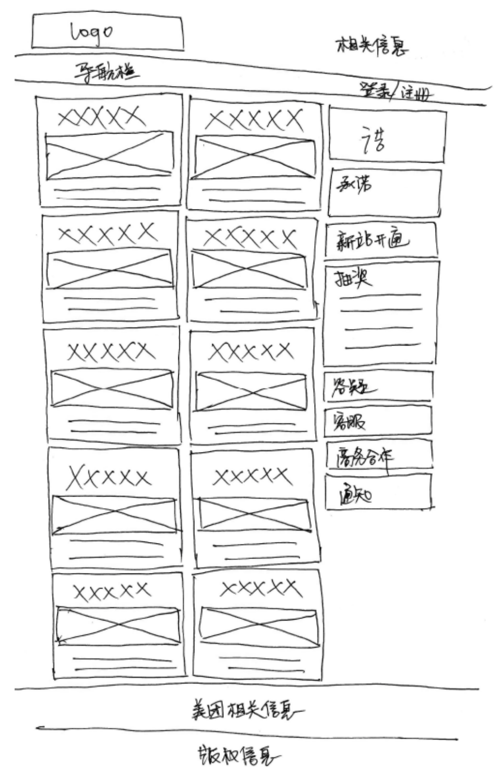


图 15.6　设计草图





通过草图对将要制作的页面进行初步分析，使整个页面大体有一个轮廓。然后就可以通过画图软件 Photoshop 设计效果图了。

第 1 步，启动 Photoshop，新建文档，设置文档大小为 1002 像素×1670 像素，分辨率为 96 像素/英寸，然后保存为“设计图.psd”。借助辅助线设计出网站的基本轮廓，如图 15.7 所示。

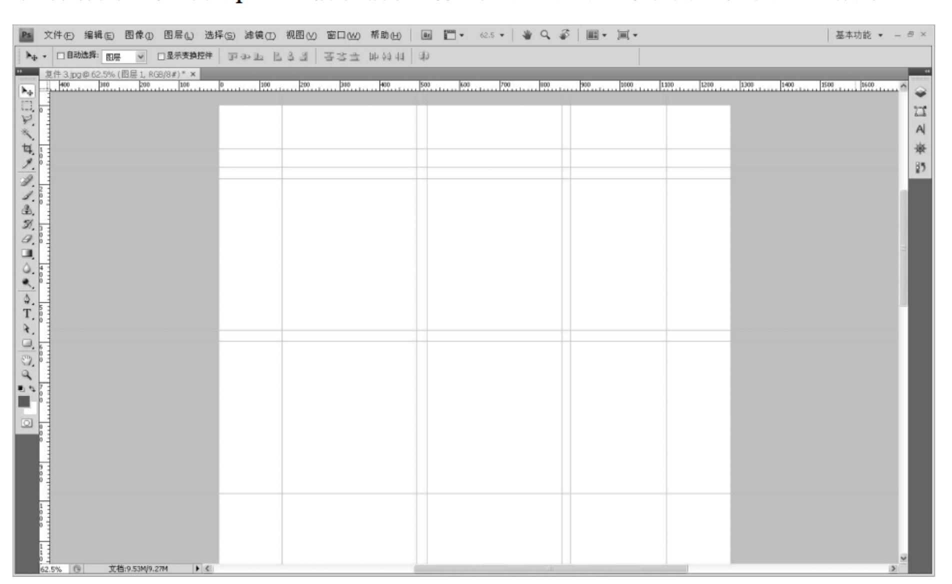


图 15.7　设置辅助线

第 2 步，在 Photoshop 中新建“线框图”图层，使用绘图工具绘制页面的基本相框和背景样式，如图 15.8 所示。

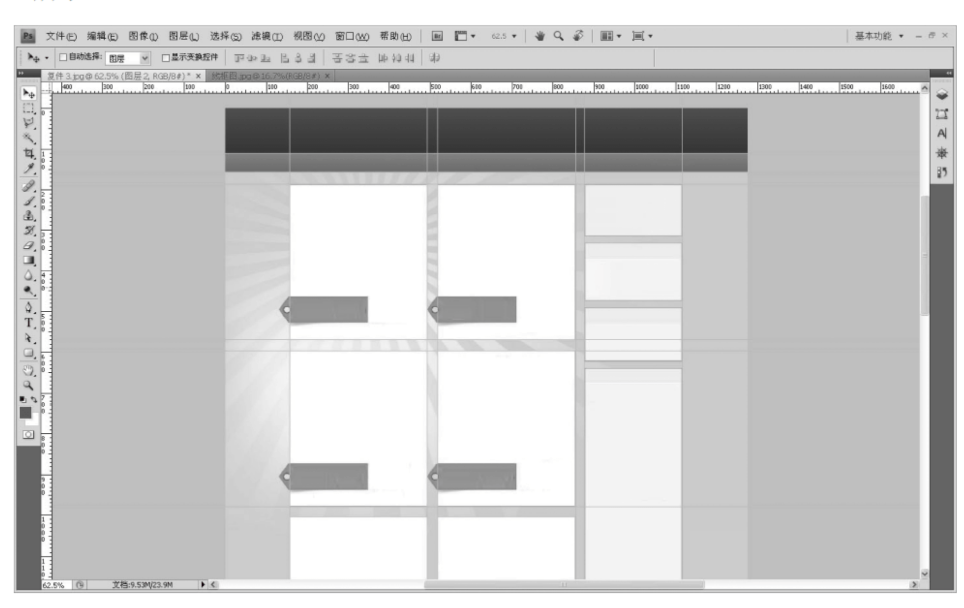


图 15.8　绘制页面线框





Note

第 3 步，在线框的基础上进一步细化栏目形态，明确栏目样式和内容，特别是重要内容的显示形式，效果如图 15.9 所示。

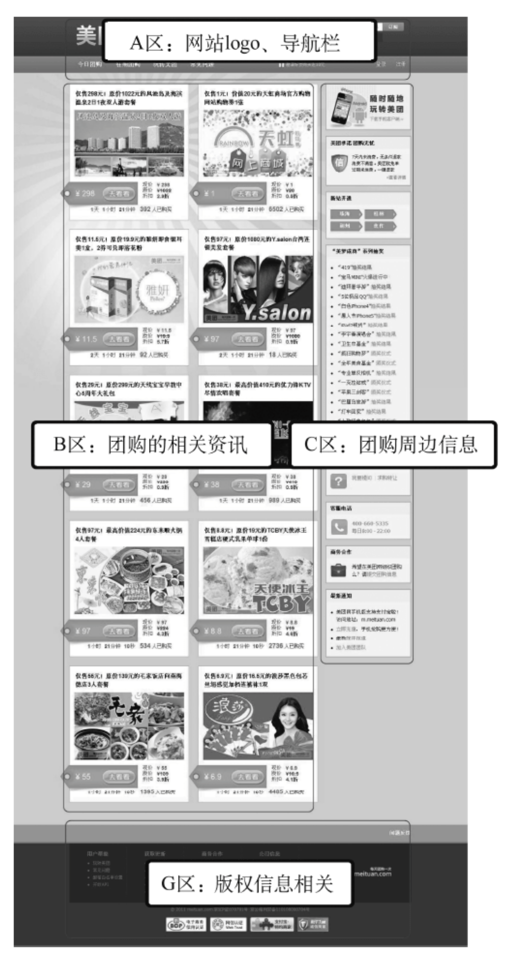


图 15.9　版块划分图

15.3　切图和输出

上面的工作完成之后，接下来就是切图了。切图的目的是要找出设计图中有用的区域，包括使用 CSS 无法实现的效果，以及可以重复显示的效果。限于篇幅，这里只讲解部分设计图。

纵观全图，看看哪些是要切下来做图片或背景的，做背景要怎么切，等等。做到心里有数，然后就可以切图了。

首先是网站的 logo 和网站头部的背景，网站的 logo 还包括其域名。广告词放到一张图里面切割下来，而背景图后期可以通过 CSS 样式设置其水平平铺显示，所以这里只要切割一小截即可，如图 15.10 所示。







图 15.10　切割网站 logo 和背景图

下面是网站的某一期的团购单元价格显示和对其相关的操作区域，价格是后台数据显示的，而后面的“去看看”按钮则连同五边形一起作为一张图片切割下来，如图 15.11 所示。



图 15.11　切割页面的背景图

其实，“去看看”只是其中一张图片，根据网站的需要，我们还要设计几张类似图片，如图 15.12 所示，然后逐张切割下来。

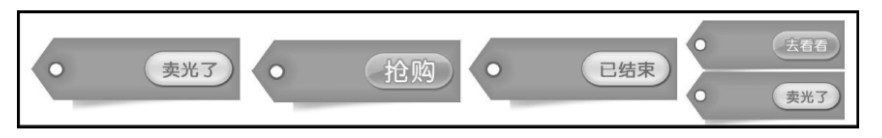


图 15.12　其他相关的切割图片

为了方便图片的管理和加载，这里把所有图片（重复利用的）都整合到一张背景透明的大图上，效果如图 15.13 所示。

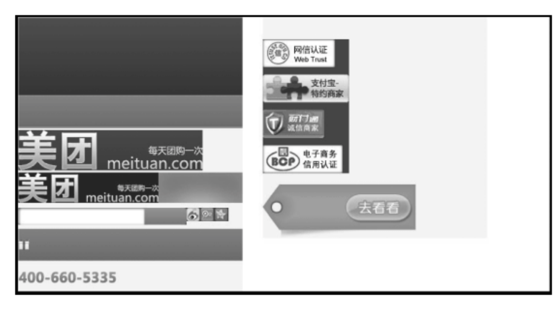


图 15.13　切割的图片整合到一张 PNG 大图上





其他边框都是简单的单色相框，效果都可以通过 CSS 实现，没有必要进行切割。

最后，把图片存储为 Web 和设备所用格式。打开存储图片的文件夹（一般以 images 命名），就可以看到网站所需要的背景图像，如图 15.14 所示。根据需要对其重命名即可。



图 15.14　切图效果

15.4　网 站 重 构

第 1 步，根据设计版块划分图的区域进行划分，启动 Dreamweaver，选择“文件”→“新建”命令，弹出“新建文档”对话框。新建立一个空白的 HTML 文档页面，并保存文件为 index.html。

第 2 步，编写 HTML 基本结构。在编写结构时，读者应该注意结构的嵌套关系，以及每级结构的类名和 ID 编号，详细代码如下。

```
<!doctype html>
<html>
<head>
<title>美团网</title>
<meta charset="utf-8">
</head>
<body class="bg-alt">
<div id="doc" >
    <div id="hdw" >
        <div id="hd" ></div>
    </div>
    <!--导航居中-->
```





```
<div id="bdw" class="bdw">
    <div id="bd" class="cf">
        <div id="index">
            <div id="content">
                <div id="index-deals">
                    <div class="secondary">
                        <div class="item odd">
                            <h1></h1>
                            <div class="cover"></div>
                            <div class="inner"></div>
                        </div>
                        ......
                    </div>
                </div>
            </div>                          <!--content 结束-->
            <div id="sidebar">
                <div id="bannerbox" class="sidebox banner"></div>
                <div class="sidebox commitment"></div>
                <div class="sidebox new-cities"></div>
                <div class="sidebox lottery"></div>
                <div class="sidebox consult"></div>
                <div class="sidebox service"></div>
                <div class="sidebox business"></div>
                <div class="sidebox notice"></div>
            </div>            <!--sidebar 结束-->
        </div>
    </div>                          <!-- bd end -->
</div>                          <!-- bdw end -->
<div id="ftw">
    <div class="contact">问题反馈</div>
    <div id="ft">
        <ul class="nav cf">
            <li class="col"></li>
            ......
            <li class="col logo"></li>
        </ul>
        <div class="copyright"></div>
        <ul class="cert cf"></ul>
```







```
            </div>              <!-- ft end -->
        </div>      <!-- ftw end -->
    </div>      <!-- doc end -->
    </body>
    </html>
```

第 3 步，简化代码，以便可以更清晰地看到整个网站的总体结构，这样到后面做网站布局的时候我们才能做到心中有数。如图 15.15 所示为几个主要模块的包含框。



图 15.15　几个主要层的包含标签

15.5　网 站 布 局

通过上面的网站重构，我们可以大致了解网站的总体结构，下面就一步步实现整个网站。

启动 Dreamweaver 软件，打开上一节中重构的网页结构文档 index.htm，然后逐步添加页面的微结构和图文信息，主要包括段落、列表、标题，以及必要的文本和图像内容，对于需要后台自动生





成的内容，可以填充简单的图文，以方便在设计时预览和测试。

第 1 步，新建样式表文档。

选择“文件”→“新建”命令，创建外部 CSS 样式表文件，保存为 style.css 文件，并存储在 images 文件夹中。然后选择“窗口”→“CSS 样式”菜单命令，打开“CSS 样式”面板，单击“附加样式表”按钮，在弹出的“链接外部样式表”对话框中，单击“浏览”按钮，找到 style.css 文件，将其链接到 index.htm 文档中，最后单击“确定”按钮，如图 15.16 所示。



图 15.16　链接外部样式表

在“代码”视图下，可以看到 Dreamweaver 自动生成的代码：

```
<link rel=stylesheet type=text/css href="images/style.css" media=all>
```

第 2 步，初始化标签样式。

将所有要用到以及即将用到的元素初始化，确保所有元素在不同浏览器下默认状态是一致的，代码如下。具体代码请查看 style.css 文件头部初始化。

```
html {padding:0px;margin:0px; }
body {padding:0px; margin:0px; }
div {padding:0px; margin:0px; }
p {padding:0px; margin:0px; }
h1 {padding:0px; margin:0px; }
ol {padding:0px; margin:0px; }
ul {padding:0px; margin:0px; }
li {padding:0px; margin:0px; }
……
```

第 3 步，分析 A 区的网站顶部<div class="hdw">的结构和样式。

该区域最外面的包含框为<div id="hdw">，设置层本身和里面的内容均水平居中，高为 157 像素，背景水平平铺，一直到浏览器的边缘。里面再写一个层<div id="hd">，宽为 960 像素。设置其堆叠顺序为 2，保证在页面的前面，相对定位，为后面的绝对定位做准备。这里要重点分析的是<a class="logo">连接标签，它以块状的方式显示，宽为 263 像素，高为 57 像素。“美团网” 这 3 个字缩进−9999 像素。这样在页面上是没办法看到的，实际上相当于把它隐藏起来。最重要的是设置背景



Note

图为 logo 的图片，也就是说网站的 logo 是以背景而不是图片的形式显示的。

“切换城市”最外面的包含层为<div class="guides">，设置其堆叠顺序为 4，也就是说显示在<div id="hd">层的上面，设置其绝对定位，并设置其定位的坐标值。里面再写一个层<div class="city">，同样为绝对定位。里面写一个<h2>，设置字体为 16 像素，颜色为白色。而“切换城市”则写在<em>标签里面，以块状的方式显示，设置字体大小为 12 像素，并设置其颜色。分析如图 15.17 所示。



图 15.17 <div class="hdw">的结构和样式

具体结构代码如下。

```
<div id=hdw style="margin:0 auto; text-align:center;">
    <div id=hd style="text-align:left;" >
        <div id=logo><a class=logo>美团网</a></div>
        <div class=guides>
            <div class=city>
                <h2 id=header-city class=fold>深圳<em><a>[切换城市]</a></em> </h2>
            </div>
        </div>
        <div id=header-subscribe-body class=subscribe></div>
        <!--放网站右边的内容-->
        <div id=header-subscribe-auto class=email-auto></div>
        <!--放选择邮件的下拉框-->
        <div style="margin:0 auto; text-align:center; width:960px; clear:both;"></div>
    </div>
    <!--网站导航及相关-->
</div>
```

下面是相关的样式代码。

```
#hdw { background: url(background.png) repeat-x; height: 157px }
/*背景图片水平平铺，一直平铺到浏览器的边缘，高为 157 像素*/
#hd { z-index: 2; position: relative; margin: 0px auto; width: 960px }
```





```
/*堆叠顺序为 2，相对定位，在页面中水平居中，宽为 960 像素*/
#logo { padding-left: 20px; padding-top: 23px }
#logo .logo { text-indent: -9999px; outline-style: none; width: 263px; display: block; background: url(background.png) #000 no-repeat 0px -160px; height: 57px }
/*文字缩进-9999 像素，相当于把文字在页面上隐藏起来，以块状的方式显示，背景图不重复，同时设置背景颜色为黑色，并设置背景图显示的位置*/
#hd .guides { z-index: 4; position: absolute; top: 39px; left: 305px }
#hd .city { position: absolute; width: 120px; top: 0px; left: 0px }
#hd .city h2 { color: #fff; font-size: 16px }
#hd .city em { outline-style: none; display: block; color: #399; font-size: 12px }
```

第 4 步，编写网站 A 区右边<div class="subscribe">层的代码。

A 区右边<div class="subscribe">层最外面的包含层为<div class="subscribe" id="header-subscribe-body">，宽为 240 像素，高为 100 像素。绝对定位，其离母层顶上边缘为 16 像素，离右边缘为 20 像素。里面分为 3 个层：第 1 个层为<div class="byemail">，宽为 240 像素，高为 24 像素。溢出来的隐藏，相对定位，设置一张背景图片。里面放一个文本域和一个按钮控件，并设置它们的相关样式。第 2 个层为<div class="followus">，设置宽度和高度，同样相对定位。“关注我们”写在<h3>里面，设置字体为 12 像素。紧跟着后面写一个< ul >，绝对定位。里面再写 3 个子标签<li>，里面分别写一个<span>标签，以块状的方式显示，宽为 20 像素，高为 19 像素，溢出来的隐藏。连接一张背景图，也就是我们看到的 3 个图标。至于里面的文字则通过设置缩进-9999 像素，把其隐藏起来。最后在它们外面都写一个<a>标签，以便可以做超链接，连接到其他页面。第 3 个层为<div class="bysms">，设置字体的颜色值为#399。里面写上 3 个<span>标签，放入相关的文字，设置内外边距等相关样式即可，最后效果如图 15.18 所示。

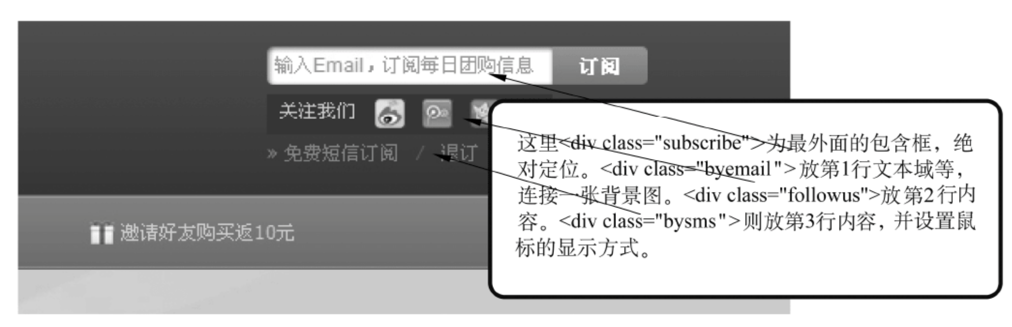


图 15.18　<div class="subscribe">层的代码

结构代码如下。

```
<div class=subscribe>
    <form id=12 action="">
        <div class=byemail>
            <input id=11 class=f-text>
```





Note

```
            <input class=commit type=submit>
        </div>
        <!--站内导航相关控件-->
        <div class=followus style="width:180px; height:26px;">
            <h3>关注我们</h3>
            <ul>
                <li><a><span class=s>美团深圳的新浪微博</span></a> </li>
                <li><a><span class=q>美团的腾讯微博</span></a> </li>
                <li><a><span class=k>美团的开心主页</span></a> </li>
            </ul>
        </div>
        <div class=bysms> <span class=sms>» 免费短信订阅</span> <span class=slash>/</span> <span
class=sms>退订</span> </div>
    </form>
</div>
```

下面是相应的样式代码。

```
#hd .subscribe { position: absolute; width: 240px; height: 100px; font-size: 12px; top: 16px; right: 20px }
/*最外面的包含层：绝对定位，字体大小为 12 像素*/
#hd .byemail { position: relative; width: 240px; background: url(background.png) 0px -270px; height: 24px;
overflow: hidden }/*相对定位，背景图片不重复，里面的内容溢出来隐藏*/
#hd .byemail .f-text { padding: 0px 5px; line-height: 22px; border-width: 0px; margin: 0px; width: 170px;
background: none transparent scroll repeat 0% 0%; height: 24px; color: #999; font-size: 12px; }
/*行高为 22 像素，边框宽度为 0 像素，没有背景图片，背景本身透明无颜色，背景图片随浏览器下拉而滚
动，背景图片重复等，里面字体大小为 12 像素*/
#hd .byemail .commit { position: absolute; border-style: none; line-height: 24px; margin: 0px; width: 60px;
display: block; background: none transparent scroll repeat 0% 0%; height: 20px; color: #fff; top: 0px; cursor: pointer;
right: 0px; font-weight: 700; _height: 24px; _padding: 2px 0 0 0 }
/*绝对定位，没有边框的样式，行高为 24 像素，以块状的方式显示，没有背景图片，背景本身透明无颜
色，背景图片随浏览器下拉而滚动，背景图片重复等，鼠标指针以手形的方式显示*/
```

上面是第 1 个层的相关样式。下面是第 2 个层的相关样式。

```
#hd .followus { position: relative; padding: 3px; margin: 6px 0px 3px; width: 158px; background: #333; height:
19px; -moz-border-radius: 3px; -webkit-border-radius: 3px; box-shadow: 1px 1px 0 #444 }
/*相对定位，兼容不同的浏览器，投影：x 轴位移 1 像素，y 轴位移 1 像素，阴影大小为 0，没有扩展，颜
色为#444（注：这是 CSS3 里面的样式）*/
#hd .followus ul { position: absolute; top: 3px; left: 68px }
#hd .followus li { float: left; margin-right: 10px }
```





```
#hd .followus span { text-indent: −9999px; width: 20px; display: block; background: url(background.png) -240px
-270px; height: 19px; overflow: hidden; cursor: pointer }
/*里面的文字缩进−9999 像素，以块状的方式显示，连接一张背景图片，溢出来的隐藏*/
#hd .followus span.q { background-position: -260px -270px }
#hd .followus span.k { background-position: -280px -270px }
#hd .followus h3 { position: absolute; color: #ccc; font-size: 12px; top: 3px; font-weight: 400; left: 8px }
```

下面是第 3 个层的相关样式。

```
#hd .bysms { color: #399; cursor: pointer }
#hd .bysms .slash {color: #666;    padding: 0px 7px 0px 5px ;}
```

第 5 步，分析 B 区<div class="item">框体的结构和样式代码。

从效果图可以看到，其实整个页面的主体内容就是由这样的一个一个具体团购单元组合起来的，只要分析完一个，其他的基本上都一样。

这里最外面的包含框为<div class="item">，宽度为 336 像素，边框是颜色值为#89b4d4 的 1 像素实线。上外边距 25 像素，背景颜色为白色，最重要的设为向左浮动。这样这些团购单元就会一个挨一个地在页面上排列起来。里面具体分 3 个层：第 1 个层为这里的标题，<h1>高为 60 像素，上、右、下、左内边距分别为 15 像素、16 像素、10 像素和 16 像素。里面字体大小为 16 像素，颜色为黑色，居左显示。第 2 个层为<div class="cover">，相对定位。里面放这个模块的大图片。第 3 个层为<div class="inner">，里面分 4 个层来完成布局，相对比较复杂，我们将在下一步进行分析。B 区框体结构如图 15.19 所示。



图 15.19　B 区<div class="item">框体结构

具体结构代码如下。




Note

```
<div class=item>
    <h1><a></a></h1>
    <!--标题-->
    <div class=cover><a><img src=""></a> </div>
    <!--大图片-->
    <div class=inner></div>
</div>
```

下面是相关的样式代码。

```
#index-deals .secondary .item { border: #89b4d4 1px solid; width: 336px; margin-bottom: 25px; background: #fff; float: left; }
/*边框是颜色为#89b4d4 的 1 像素实线，宽为 336 像素，背景颜色为白色，向左浮动*/
#index-deals .secondary h1 { text-align:left; padding:15px 16px 10px 16px; height: 60px; font-size: 16px; }
/*字体 16 像素，居左对齐，上、右、下、左内边距分别为 15 像素、16 像素、10 像素和 16 像素*/
#index-deals .secondary h1 a { color: #000 }
#index-deals .secondary h1 a:hover { color: #399 }
#index-deals .secondary .cover { position: relative }
```

第 6 步，分析<div class="inner">的框体结构。

接着上一步对第 3 个层<div class="inner">进行分析。前面也说过，这里面还分 4 个层，其中难点便是第 1 个层<div class="deal-buy">。从效果图看到，“去看看”这一图片是超出整个单元最外面的包含层<div class="item">的，到底是怎么实现的呢？

先设置层<div class="deal-buy">为相对定位，里面再写 1 个层<div class="deal-price-tag-open">，设置宽为 237 像素，高为 76 像素，为绝对定位，其离母层<div class="deal-buy">左边缘为−27 像素，而左内边距为 50 像素。为了保证层本身在页面的前面，设堆叠顺序为 1。这样，最后设置一张背景图，即可得到背景图跑出外面的效果。总的来说这里是通过两步来完成这样的效果的：先用绝对定位把层写到母层的包含框外面，然后用堆叠顺序保证这个层在母层的上面。

通过上一步，背景图就设置好了，然后就是安排里面的内容了。这里先写一个<p class="deal-price">，宽为 100 像素，高和行高为 64 像素，而左内边距为 35 像素。同样设置绝对定位。其离母层左边缘为−24 像素，设置堆叠顺序为 2，这样就会在层<div class="deal-price-tag-open">的上面。这样内容的层也做好了。至于里面的价格数值就写在<strong>里面。设置左浮动，字体为白色，大小为 22 像素。

而我们知道“去看看”这几个字其实是一张背景图，那怎么写代码才会让其只要被点击就会产生相应操作的效果呢？先写一个<span>标签，同样用绝对定位设置这个标签的位置。里面再写一个<a>连接标签，以块状的方式显示。宽为 97 像素，高为 41 像素。上外边距为 12 像素。这样当鼠标指针移到背景图“去看看”这一区域周围时，便可做相应的操作了。

第 2 个层为<table class="discount">，里面分 3 行 6 个单元格，写上相关样式即可。





第 3 个层为<div class="deal-box deal-timeleft deal-on">，设为左浮动，里面写一个<ul>，其子标签<li>以行内的方式显示，这样便可以让里面的内容在一条线上显示了。

第 4 个层为<div class="deal-box deal-status deal-status-open">，同样设为左浮动，设置内外边距等相关样式，最后如图 15.20 所示。

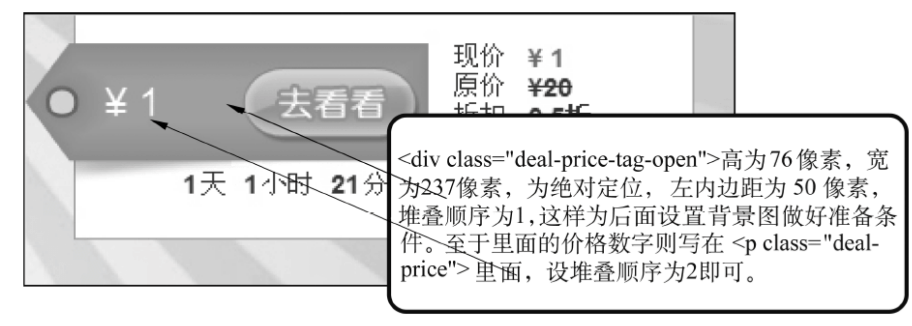


图 15.20　<div class="inner">的框体结构

结构代码如下。

```
<div class=inner>
    <div class=deal-buy>
        <div class=deal-price-tag-open></div>
        <p class=deal-price><strong></strong> <span><a></a></span> </p>
    </div>
    <table class=discount>
        <tr><th>现价</th><td class=price><strong></strong></td></tr>
        <tr><th>原价</th><td><del><strong></strong></del></td> </tr>
        <tr><th>折扣</th> <td><strong></strong></td></tr>
    </table>
    <div class="deal-box deal-timeleft deal-on">
        <ul>
            <li><span></span></li>
        </ul>
    </div>
    <div class="deal-box deal-status deal-status-open">
        <p class=deal-buy-tip-top><strong>6502</strong>  人已购买</p>
    </div>
</div>
```

下面是一部分样式代码。

第 1 个层<div class="deal-buy">的样式代码如下。




```
.item .inner { text-align:left; }
#index-deals .secondary .deal-buy { position: relative }
#index-deals .secondary .deal-price-tag-open { text-align:left; z-index: 1; position: absolute; padding-left: 50px; width: 237px; background: url(bg-deal-see-s.png) no-repeat 0px 0px; height: 76px; left: -27px; }
/*堆叠顺序为 1，绝对定位，左内边距为 50 像素，宽为 237 像素，背景图不重复*/
#index-deals .secondary .deal-price { z-index: 2; position: absolute; line-height: 64px; padding-left: 35px; width: 100px; height: 64px; left: -24px }
/*堆叠顺序为 2，绝对定位，行高和高为 64 像素，左内边距为 35 像素，离母层左边-24 像素*/
#index-deals .secondary .deal-price strong { float: left; color: #fff; margin-left: 5px; font-size: 22px; font-weight: normal }
/*左浮动，字体颜色为白色，大小为 22 像素，左外边距为 5 像素*/
#index-deals .secondary .deal-price span { z-index: 2; position: absolute; color: #fff; top: 0px; left: 113px }
#index-deals .secondary .deal-price span a { line-height: 41px; margin-top: 12px; outline-width: 0px; width: 97px; display: block; height: 41px }
/*行高和高都为 41 像素，外上边距为 12 像素，以块状的方式显示*/
```

以下为第 2 个层<table class="discount">的样式代码。

```
#index-deals .secondary .discount { margin: 0px 0px 10px 205px; width: 120px; color: #333; font-size: 14px }/*上、右、下、左外边距分别为 0 像素、0 像素、10 像素和 205 像素，宽为 120 像素*/
#index-deals .secondary .discount th { text-align: left; padding: 0px; }
#index-deals .secondary .discount td { text-align: left; padding: 0px; }
#index-deals .secondary .discount th { width: 40px; font-weight: normal }
#index-deals .secondary .discount td { width: auto }
#index-deals .secondary .discount .price { color: #c33 }
```

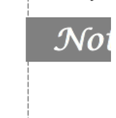
以下为第 3 个层<div class="deal-box deal-timeleft deal-on">的样式代码。

```
#index-deals .secondary .deal-timeleft { text-align: right; width: 184px; padding-right: 0px; float: left; height: 22px; color: #333 }
#index-deals .deal-timeleft ul { display: inline }
#index-deals .deal-timeleft li { display: inline }
#index-deals .deal-timeleft li span { padding: 0px 2px; font-weight: bold; }
```

以下为第 4 个层<div class="deal-box deal-status deal-status-open">的样式代码。

```
#index-deals .secondary .deal-status { padding: 8px; width: 120px; float: left; color: #000; margin-left: 5px; }/*内边距为 8 像素，宽为 120 像素，左浮动，左外边距为 5 像素*/
#index-deals .secondary .deal-buy-tip-top { line-height: 1; font-size: 14px; font-weight: normal }
#deal-intro .deal-buy-tip-top strong { color: #c33 }
```





第 7 步，编写 C 区 <div class="sidebox commitment">的结构和样式。

这里最外面的包含框为<div class="sidebox commitment">，边框是颜色为#89b4d6 的 2 像素的实线，并且下外边距为 16 像素。标题“美团承诺 团购无忧”写在<h3>里面，行高和高为 32 像素，这样可以垂直居中。底边框为 1 像素的白色实线。上、右、下、左 4 个边的内边距分别为 12 像素、12 像素、0 像素和 12 像素。

<div class="sideinner">则负责放这一模块里面的内容。最小高度为 48 像素，高度设为自适应，这样可以保证本层就算没有内容也有 48 像素的高度。同时，上、右、下、左 4 个边的内边距分别为 12 像素、12 像素、0 像素和 12 像素。而上边框为 1 像素的实线。里面先写一个<ul>，设置一张背景图，即从效果图看到的有个“信”字的图标。左内边距为 60 像素，这样文字便会显示在图标的右边。后面的 3 行文字分别写在 3 个子标签<li>里面。

至于“查看详情”要写在<p>里面，设置居右对齐，文字便在右边显示，效果如图 15.21 所示。

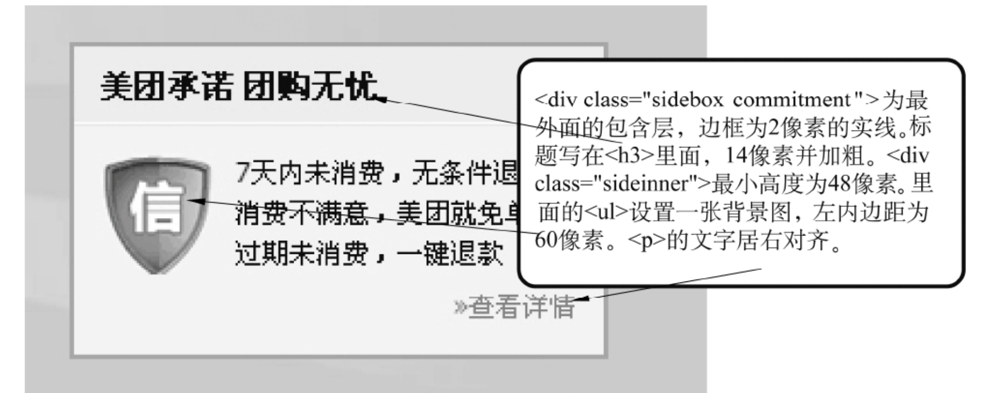


图 15.21　<div class="sidebox commitment">的结构

下面是这个区域总体框架的结构代码。

```
<div class="sidebox commitment">
    <h3>美团承诺 团购无忧</h3>
    <div class=sideinner>
        <ul>
            <li></li>
            <!--放 3 行文字-->
        </ul>
        <p><a>查看详情</a></p>
    </div>
</div>
```

具体样式代码如下。







```
    #sidebar .sidebox { border: #89b4d6 2px solid; padding-bottom: 12px; background-color: #eff8ff; margin-bottom: 16px; }
    /*边框是颜色为#89b4d6 的 2 像素的实线，下内边距为 12 像素，下外边距为 16 像素*/
    #sidebar .sidebox h3 { border-bottom: #fff 1px solid; padding: 0px 12px; line-height: 32px; background: #eee; height: 32px; font-size: 14px; _padding-top: 2px }
    /*底边框为 1 像素的白色实线，行高和高为 32 像素，字体大小为 14 像素*/
    #sidebar .sideinner { padding:12px 12px 0px 12px; min-height: 48px; height: auto !important; border-top: #eaecec 1px solid; }
    #sidebar .commitment ul { padding-left: 60px; background: url(background.png) no-repeat 0px -460px; font-size: 12px }/*左内边距为 60 像素，背景图片不重复，字体大小为 12 像素*/
    #sidebar .commitment p { text-align: right; padding:6px 0px 0px 0px; font-size: 12px; }
```

上面分析的是 C 区其中一个栏目的结构和样式代码，C 区中其他的栏目也可以以此类推，篇幅有限，所以就不再一一分析了。

第 8 步，分析 D 区里面<ul class="nav cf">的结构和样式。

D 区为网站的版权信息区域，而我们要分析的是<ul class="nav cf">这一层。从效果图得知，里面的内容可以分为 5 个部分，也就是说先写 5 个子标签<li class="col">，设置高度为 140 像素，宽度为 110 像素。右边框为 1 像素灰色虚线，即从效果图看到的右边虚线分隔线。设定内外相关边距等。以行内的方式显示。当然最重要的是左浮动，这样 5 个<li>才能在一个水平线上并排显示。

同时，前面 4 个结构是一样的，只要分析其中一个，其他的可以以此类推。先在里面写一个<h3>，放相关的小标题，如第一个放“用户帮助”，宽为 110 像素，字体大小为 14 像素。接着写一个<ul class="sub-list">，同样宽为 110 像素，上外边距为 5 像素。接着里面有多少行文字就写多少个<li>，如第一个有 4 行文字，则写 4 个<li>，设其行高为 20 像素。前面的装饰为圆点，即 list-style-type:disc，并且在里面显示，即 list-style-position : inside。

最后一个<li>是网站的 logo。其实现办法是在里面写一个<a>标签，宽为 200 像素，高为 43 像素，以块状的方式显示，并且溢出来的隐藏。通过缩进−9999 像素，把文字隐藏起来，再连接一张 logo 背景图片即可，同时用外边距调节一下其显示的位置，如图 15.22 所示。

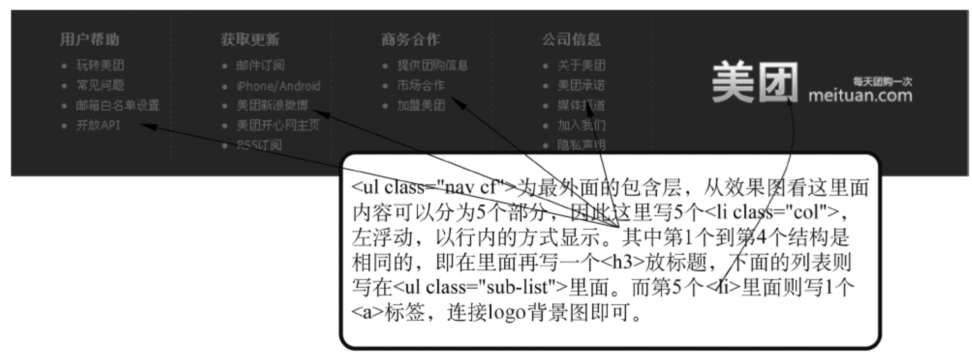


图 15.22　<ul class="nav cf">的结构分析



结构代码如下。

```
<ul class="nav cf">
    <li class=col>
        <h3>用户帮助</h3>
        <ul class=sub-list>
            <li><a>玩转美团</a></li>
            <li><a>常见问题</a></li>
            <li><a>邮箱白名单设置</a></li>
            <li><a>开放 API</a> </li>
        </ul>
    </li>
    <li class=col></li>     <!--获取更新-->
    <li class=col></li>     <!--商务合作-->
    <li class=col></li>     <!--公司信息-->
    <li class="col logo"><a>美团网</a> </li>
</ul>
```

从上面的结构代码和说明图可以看到，其结构是很清楚的，下面再写出样式代码即可。

```
.cf { zoom: 1 }
.cf:after { display: block; height: 0px; visibility: hidden; clear: both; overflow: hidden; content: '' }
/*这里为.cf 写了一个伪类*/
#ft .nav { background: #222; color: #666 }
#ft .nav li.col { margin: 10px 0px; width: 110px; display: inline; float: left; height: 140px; border-right: #333 1px dashed; padding: 10px 0px 0px 50px; }
/*上下外边距为 10 像素，左右为 0，以行内的方式显示，左浮动，右边框为 1 像素灰色虚线*/
#ft .nav li.logo { border-style: none; }
#ft .nav li.logo a { text-indent: -9999px; margin: 40px 0px 0px 10px; width: 200px; display: block; background: url(background.png?v=20110413) 0px -220px; height: 43px; overflow: hidden }
/*文字缩进-9999 像素，宽为 200 像素，以块状的方式显示，背景图不重复，溢出来的隐藏*/
#ft .nav h3 { color: #666; font-size: 14px; width:110px; text-align:left; }
#ft .sub-list { margin-top: 5px; font-size: 12px; width:110px; }
#ft .sub-list li { list-style-position: inside; line-height: 20px; list-style-type: disc; text-align:left; }
/*列表的样式类型为原型，显示的位置在里面，行高为 20 像素，居左对齐*/
```

第 9 步，完成网站其他地方页面的设计。

篇幅有限，上面几步只是对网站的重点和要点做了分析，只要把上面的内容分析透彻，剩下的相信读者自己也可以完成，所以这里就不再做更多分析了。



掘客类型网站的布局与设计

掘客类网站其实就是一个文章投票评论站点，它结合了书签、博客、RSS 以及无等级的评论控制。这类网站的独特性就在于它没有职业网站编辑，编辑全部来源于用户。用户可以随意提交文章，然后由阅读者来判断该文章是否有用，收藏文章的用户人数越多，说明该文章越有热点。用户认为这篇文章不错，收藏或点击量到一定的程度，那么该文章就会出现在首页或者其他页面上。

下面是一些典型的掘客类网站：新蓝房产品地客（http://fangchan.cztv.com/，如图 16.1 所示）、至酷掘客网（http://dig.gku.cn/）、奇客发现（http://www.diglog.com/，如图 16.2 所示）、掘客手机论坛（http://www.ppc-rom.com/）、天水掘客网（http://www.dig7.cn/）和校园掘客（http://www.9idig.com/）。



图 16.1　品地客　　　　图 16.2　奇客发现





16.1 产 品 策 划

掘客网站与博客一样，网站的主体内容都是以文章为主的文字信息，故设计时应尽可能多地考虑到这样的特点，怎样把文章分门别类地安排好展示给大众才是最重要的。

既然跟博客是同一类别的网站，因此基本上都是以两列多行的方式布局，如图 16.3 所示，当然 1 列多行也是不错的选择，如图 16.4 所示，还可以是 3 列多行，如图 16.5 所示。

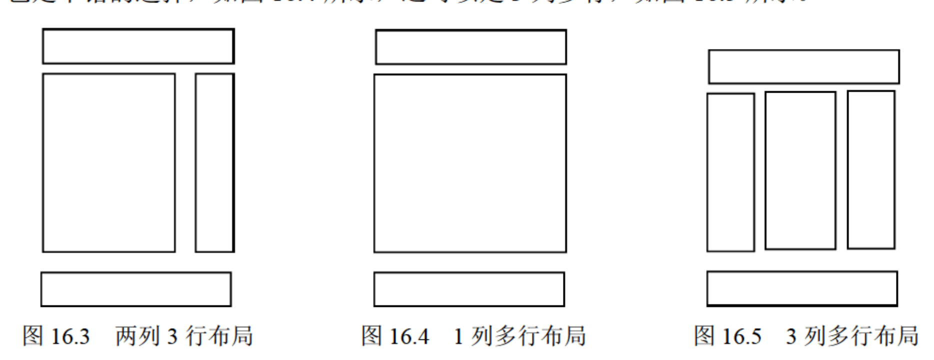


图 16.3 两列 3 行布局　　图 16.4 1 列多行布局　　图 16.5 3 列多行布局

本案例单从设计的角度来说也能让人感觉到是 Web 2.0 网络时代的产物。

首先是网站头部，用了一幅很有时代气息同时又很有漫画色彩的背景图，再者 logo 也设计得很有诗意。加上右边网站导航等相关操作的设计。整个头部看似出人意料却又在情理之中。

再认真分析，整个头部只有 3 种颜色：黑色、白色和灰色。但通过相互之间的搭配和灰色不同程度的提炼使用，设计得却如此出色和恰如其分。由此得出设计不在于颜色使用的多少，而在于对每一种颜色的使用和把握的能力。

从整个页面来看，页面以灰色和白色为主色调，这也很好地暗示给浏览者这是一个理性和追求真知灼见的地方，正好表现了网站的主旨思想。

为了避免过于呆板，网站还用了几个其他色相的颜色作为点缀。在“掘客热点”和下面主要内容区域每个栏目的投票票数都用了很鲜明的深蓝色和青蓝色。这些作为网站很重要的点，颜色虽鲜明，所占空间却很少，不但不会使网站显得浮躁，相反在大面积灰色背景的衬托下，整个网站都显得沉稳又有灵性。

另外，网站的布局更是有目共睹，3 行两列的排版方式，更巧的是为了稍微打破这种千篇一律的格局，在网站主体内容开始的头部就用了类似选项卡的设计。蓝色还延伸到主体内容区域里面来。可见网站的布局也是匠心独运的。

16.2 画板和设计

根据策划的基本思路，在产品设计时，本案例在准确表达内容的基础上可以适当展示一些 Web2.0





Note

网络时代的特征，整个网站包括如图 16.6 所示的网站 logo、导航栏和站内搜索、栏目切换、网站主体内容、分类、文章队列、最新评论、热门标签、网站相关和版权信息等版块内容。



图 16.6　网站效果图

根据这些模块，初步在稿纸上画出页面布局的草图，如图 16.7 所示。

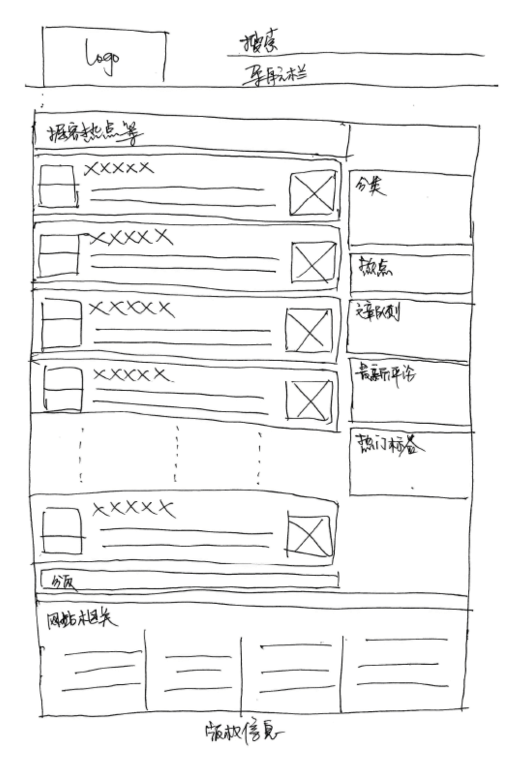


图 16.7　网站草图





通过草图对将要制作的页面进行初步分析，使整个页面大体有一个轮廓。现在就可以通过 Photoshop 设计效果图了。

第 1 步，启动 Photoshop，新建文档，设置文档大小为 1002 像素×1200 像素，分辨率为 96 像素/英寸，然后保存为“设计图.psd”。借助辅助线设计出网站的基本轮廓，如图 16.8 所示。

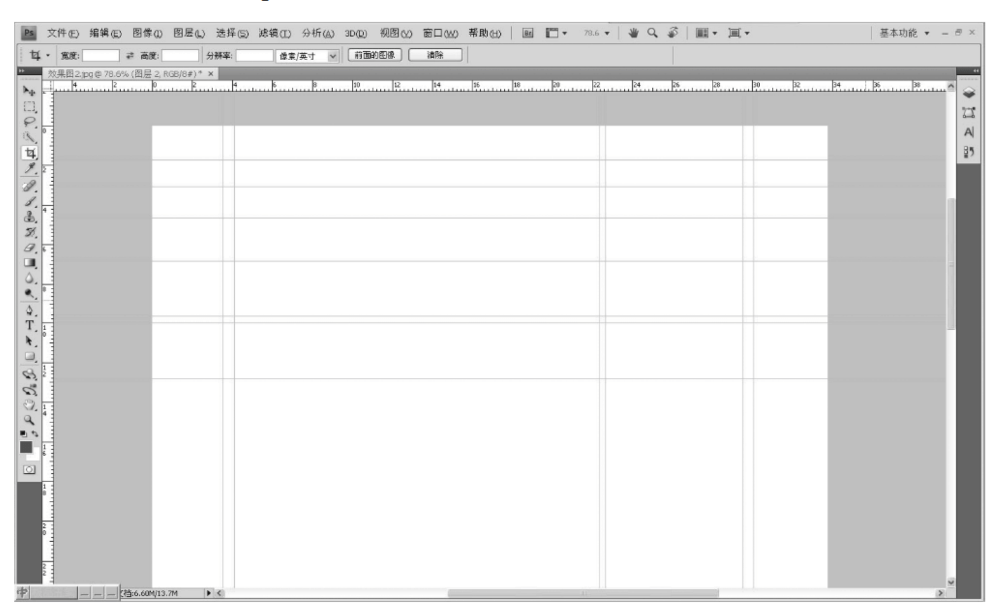


图 16.8　设置辅助线

第 2 步，新建“线框图”图层，使用绘图工具绘制页面的基本相框和背景，如图 16.9 所示。

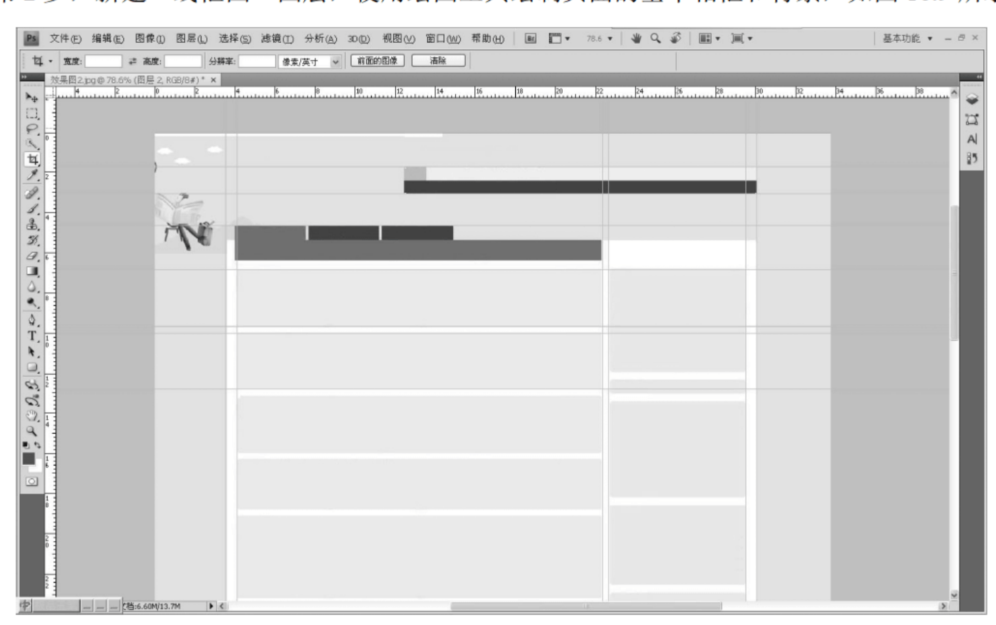


图 16.9　绘制页面线框





Note

第 3 步，在线框的基础上进一步细化栏目形态，明确栏目样式和内容，特别是重要内容的显示形式，如图 16.10 所示。



图 16.10　版块划分图

16.3　切图和输出

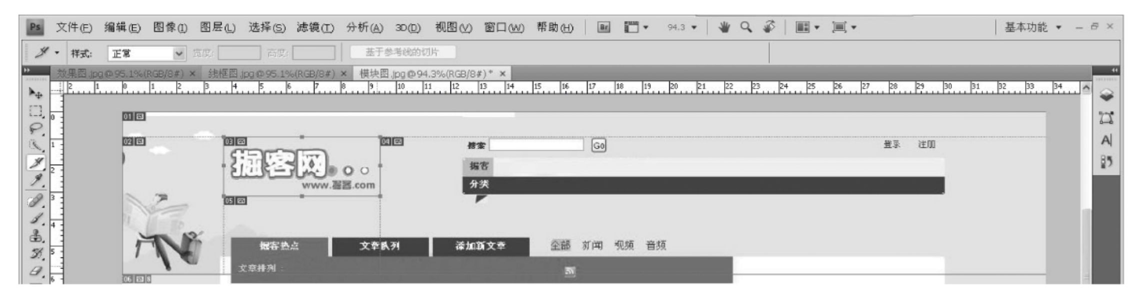
上面的工作完成之后，接下来就是切图了。切图的目的是找出设计图中有用的区域，包括使用 CSS 无法实现的效果和可以重复显示的效果。

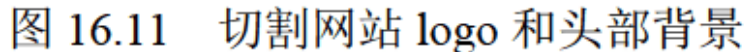
第 1 步，纵观全图，看看哪些是要切下来做图片或背景的，做背景要怎么切，等等。做到心里有数，然后就可以切图了。首先是网站的 logo，这里切割为背景透明的 PNG 图片。同时把网站头部





很大、很潮的 Logo 作为一张大背景图也切割下来，如图 16.11 所示。




图 16.11　切割网站 logo 和头部背景

第 2 步，网站主体区域，即文章栏目左边的投票数和投票操作的背景图，切割如图 16.12 所示。

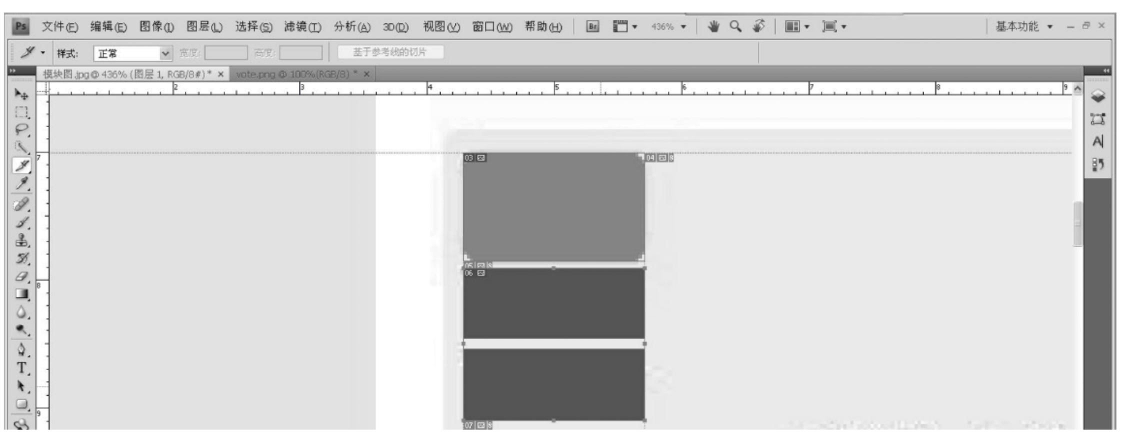


图 16.12　切割投票数和投票操作的背景图

第 3 步，网站侧栏浅蓝色背景图因为有圆角效果，单纯用样式实现不了，故切割下来，如图 16.13 所示。

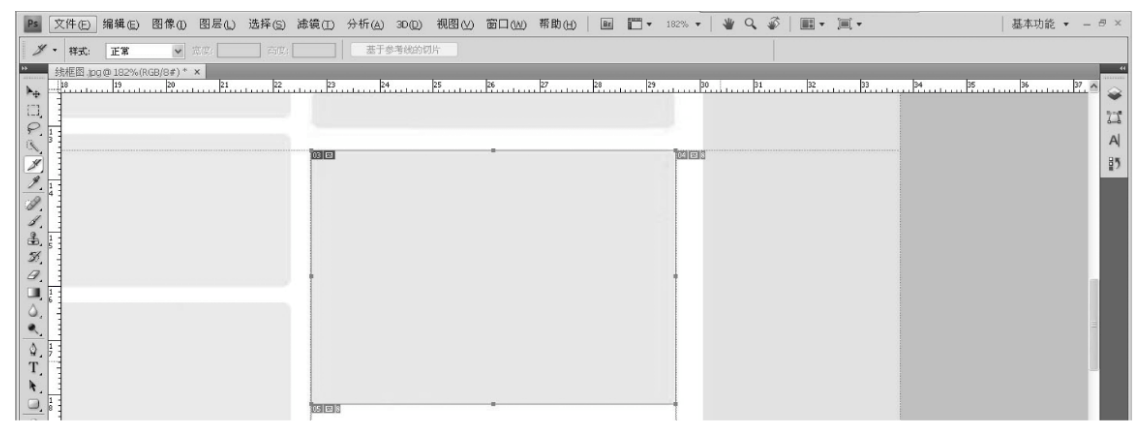


图 16.13　网站侧栏的背景图片



Note

剩下大部分边框都是简单的单色相框，可以通过 CSS 实现，故没有必要进行切割。

第 4 步，将图片存储为 Web 和设备所用格式。具体操作方法读者可查看前面章节的相关内容，这里不再重复阐述。

第 5 步，在 images 文件夹中可以看到所需要的背景图像，如图 16.14 所示。再根据需要将其重命名即可。



图 16.14　切图效果

16.4　网 站 重 构

第 1 步，根据设计版块划分图的区域进行划分，启动 Dreamweaver，选择“文件”→“新建”命令，弹出“新建文档”对话框。新建立一个空白的 HTML 文档页面，并保存文件为 index.html。

第 2 步，编写 HTML 基本结构。在编写结构时，读者应该注意结构的嵌套关系，以及每级结构的类名和 ID 编号，详细代码如下。



```
<!doctype html>
<html>
<head>
<title>掘客网</title>
<meta charset="utf-8">
</head>
<body>
<div id="header">
    <h2 id="logo"></h2>
```







```
        <div id="topnav"></div>        <!-- end: id="topnav" -->
        <div id="nav"></div>
    </div>                <!-- header end -->
    <div id="wrap">
        <div class="navigation-sub">
            <ul class="clearfix secondary"></ul>
        </div>
        <div id="content-wrap2">
            <div id="sidebar">
                <div class="featurebox">
                    <div class="tlb">分类</div>
                    <div id="s1"></div>
                </div>
                <div class="featurebox">
                    <div class="tlb">今日热点</div>
                    <div id="sstop"></div>
                </div>
                <div class="featurebox">
                    <div class="tlb">文章队列</div>
                    <div id="ssstories"></div>
                </div>
                <div class="featurebox">
                    <div>最新评论</div>
                    <div id="latcomments"></div>
                </div>
                <div class="featurebox">
                    <div class="tlb">热门标签</div>
                    <div id="s2"></div>
                </div>
            </div>                <!-- sidebar end -->
            <div id="contentbox">
                <div id="xnews-5" class="news-summary">
                    <div>
                        <div class="roundbarfg">
                            <ul class="news-upcoming"></ul>
                            <div class="top"></div>
                        </div>
                    </div>
```






```
</div>                    <!-- 书签 | 书签.com end -->
<div id="xnews-6" class="news-summary">
    <div class="roundbarfg">
        <ul class="news-upcoming"></ul>
        <div class="top"></div>
    </div>
</div>                    <!-- 巴黎之夜 end -->
<div id="xnews-7" class="news-summary">
    <div class="roundbarfg">
        <ul class="news-upcoming"> </ul>
        <div class="top"></div>
    </div>
</div>                    <!-- iphone 最牛的用法_在线视频观看 end -->
<div id="xnews-8" class="news-summary">
    <div class="roundbarfg">
        <ul class="news-upcoming"></ul>
        <div class="top"></div>
    </div>
</div>                    <!-- 两拓合并加大中方铁矿石谈判劣势 end -->
<div id="xnews-9" class="news-summary">
    <div class="roundbarfg">
        <ul class="news-upcoming"></ul>
        <div class="top"></div>
    </div>
</div>          <!-- 刘德华半空献唱贺回归 吊钢丝离地面八层楼高 end -->
<div id="xnews-10" class="news-summary">
    <div class="roundbarfg">
        <ul class="news-upcoming"></ul>
        <div class="top"></div>
    </div>
</div>                   <!--狙击老公花心的最佳对策 end -->
<div id="xnews-11" class="news-summary">
    <div class="roundbarfg">
        <ul class="news-upcoming"></ul>
        <div class="top"></div>
    </div>
</div>                  <!--分析称人民币贬值既是经济牌又是政治牌 end -->
<div id="xnews-12" class="news-summary">
    <div class="roundbarfg">
```







```
                        <ul class="news-upcoming"></ul>
                        <div class="top"></div>
                    </div>
                </div>          <!--中国上海市政府在英国举办大型金融人才招聘会   end -->
                <div class="pagination"></div>
            </div>              <!-- contentbox end -->
        </div>                  <!-- content-wrap end -->
    </div>                          <!-- wrap end -->
    <div id="footer">
        <div class="footer-contents">
            <dl class="last"> </dl>
        </div>
    </div>
    <div class="copyright"></div>
    </body>
    </html>
```

再简化一下代码，以便可以更清晰地看到整个网站的总体结构，到后面做网站布局的时候我们就能做到心中有数，如图 16.15 所示。

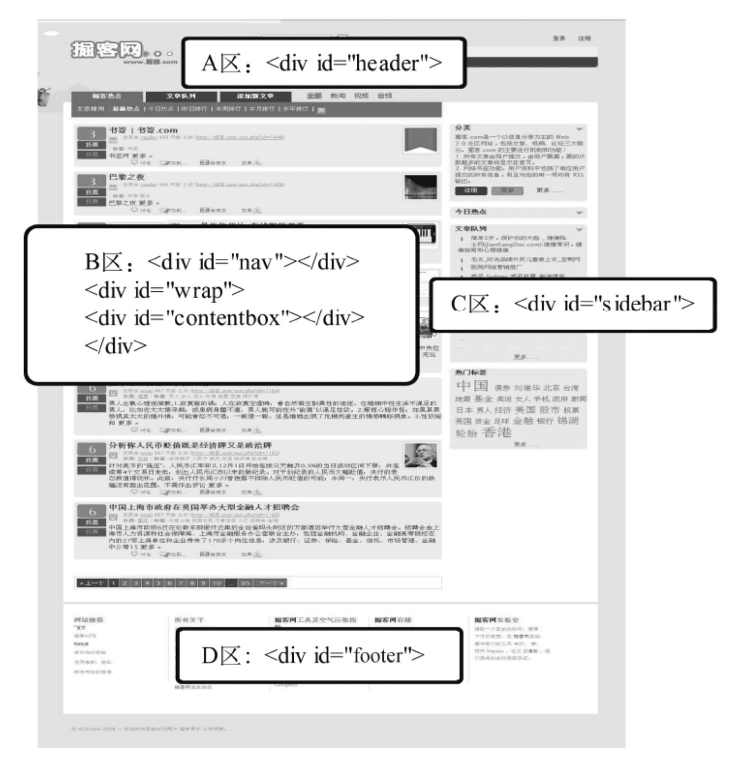


图 16.15　几个主要层的包含标签







16.5 网站布局

通过上面的网站重构，我们大致清楚网站的总体结构了，下面就一步步实现整个网站。

启动 Dreamweaver 软件，打开上一节中重构的网页结构文档 index.htm，然后逐步添加页面的微结构和图文信息，主要包括段落、列表、标题，以及必要的文本和图像内容，对于需要后台自动生成的内容，可以填充简单的图文，以方便在设计时预览和测试，等设计完毕后，再进行清理，留待程序员添加后台代码。

第 1 步，新建样式表文档。

选择“文件”→“新建”命令，创建外部 CSS 样式表文件，保存为 style.css 文件，存储在 images 文件夹中。选择“窗口”→“CSS 样式”菜单命令，打开“CSS 样式”面板，单击“附加样式表”按钮，在弹出的“链接外部样式表”对话框中，单击“浏览”按钮，找到 style.css 文件，将其链接到 index.htm 文档中，最后单击“确定”按钮。接着在页面头部添加如下代码，导入外部样式表。

```
<link rel=stylesheet type=text/css href="images/style.css" media=all>
```

第 2 步，初始化标签样式。

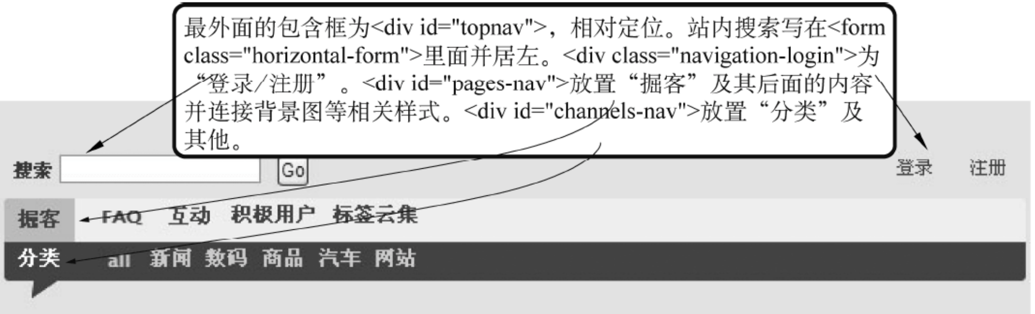
在新建的外部样式表中，将要用到的所有元素初始化，确保所有元素在不同浏览器下默认状态是一致的。代码如下，具体代码请查看 style.css 文件头部初始化。

```
* { margin: 0px; padding: 0px; }
html { min-height: 100%; height: 100% }
* html body * { overflow: visible }
* html iframe { overflow: auto }
* html frameset { overflow: hidden }
body { margin: 0px auto; font: 12px arial; background: #ebebeb; color: #414141; padding: 0px }
img { border: 0px solid; }
……
```

第 3 步，分析 A 区的网站顶部<div id="topnav">的结构（见图 16.16）和样式。

这是网站头部的右半部分，主要是站内搜索及网站导航的相关内容。首先其最外面的包含框为<div id="topnav">，相对定位，左外边距为 335 像素。里面主要分为 4 个部分。第 1 部分为<form class="horizontal-form">，上、右、下、左 4 边的内边距分别为 35 像素、0.5 像素、0.8 像素、0.5 像素，高为 20 像素。里面放相关的文字和控件并设置相对应的样式即可。第 2 部分为<div class="navigation-login">，绝对定位，宽 290 像素，右浮动。里面写上“登录”“注册”并设置相应样式。第 3、4 部分相对比较复杂，放到下一步分析。





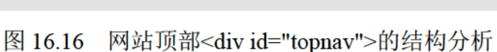


图 16.16　网站顶部<div id="topnav">的结构分析

结构代码如下。

```
<div id=topnav>
    <form id=header-form-search class=horizontal-form>
        <label><a>搜索</a></label>
        <input id=header-search-keys class=form-text/>
        <input class=button-01-xs/>
    </form>
    <div class=navigation-login>
        <ul id=header-login-links>
            <a class="join 0"><span style="font-size: 12px">登录 </span></a> <a class="join is-signed last 1"><span style="font-size: 12px">注册 </span></a> </ul>
    </div>
    <div id=pages-nav class=navigation-services></div>    <!--掘客及其他-->
    <div id=channels-nav class=navigation-channels></div>    <!--分类及其他-->
</div>
```

对应的样式代码如下。

```
#topnav { position: absolute; text-align: right; top: 10px; right: 10px }
#topnav { position: relative; text-align: left; margin-left: 335px; top: auto; right: auto; padding: 0px }
/*相对定位，左外边距为 335 像素，文字居左对齐*/
#topnav div#channels-nav { padding-bottom: 9px; background: url(logo-tail.png) no-repeat 1.5em 100% }
form.horizontal-form { padding: 0.8em 0.5em;}
form.horizontal-form * { display: inline }
form.horizontal-form input { line-height: 1.4em; margin: 0em 0.5em 0em 0em }
div.navigation-login { position: absolute; width: 290px; float: right; top: 35px; right: 0px }
/*绝对定位，离外边框上边距离为 35 像素，左边则为 0 像素，宽为 290 像素，向右浮动*/
div.navigation-login ul { padding: 0px; margin: 0px; display: inline; float: right; }
div.navigation-login ul li { padding: 2px 5px; line-height: 1em; list-style-type: none; margin: 0px; display: inline;
```



```
float: left; font-size: 91.67%; border-right: #414141 1px solid;}
    /*行高为 1em，以行内的方式显示，左浮动，右边框为 1 像素宽度的灰色实线*/
    div.navigation-login ul li.join { font-weight: bold }
    div.navigation-login ul li.last { border-right: 0px }
```

第 4 步，分析<div class="navigation-services">的结构和样式。

接着上一步的分析，第 3 部分为<div class="navigation-services">，相对定位，以块状的方式显示，里面的内容有多出来的隐藏，背景颜色的值为#eff0f0。里面先写一个<span class="corner-top">，堆叠顺序为 10，绝对定位，高为 3 像素，宽为 100%，连接一张背景图。因为 CSS 无法实现圆角效果，故这里通过一张背景图来实现。然后再写一个<a class="home">连接标签。以块状的方式显示，左浮动，背景颜色为深灰色（#bbbdbf），里面输入“掘客”并加粗显示。最后再写一个<ul id="header-user-links">，里面写 4 个子标签<li>。以行内方式显示，向左浮动等。

第 4 部分为<div class="navigation-channels">，相对定位，宽为 100%，以块状的方式显示。底内边距为 9 像素，并连接一张背景图，即效果图中最下面的方向向下的三角形。里面先写一个<ul class="primary-links">，设置向左浮动。再写上 7 个子标签<li>，行高为 1em，以行内的方式显示，向左浮动，字体加粗显示。最后写上<span class="corner-bottom">，里面连接一张背景图，实现圆角效果。最后分析如图 16.17 所示。

掘客 FAQ 互动 积极用户 标签云集
分类 all 新闻 数码 商品 汽车 网站

先写一个<span class="corner-top">连接一张背景图实现圆角效果。再写上<div id="pages-nav">放置“掘客”及其后面的内容并设置背景颜色为浅灰色等相关样式。<div id="channels-nav">放置“分类”及其他，设置背景颜色为深灰色等相关样式即可。最后通过< span class="corner-bottom">连接背景图实现圆角效果。

图 16.17　<div class="navigation-services">的结构分析

对应的结构代码如下。

```
    <div id=pages-nav class=navigation-services> <span class=corner-top><span></span></span> <a class=home>掘客</a>
        <ul id=header-user-links>
            <li class="first tool 0"><a class="tool 0">faq</a></li>
            <li class="tool 1"><a class="tool 1"><span>互动 </span></a></li>
            <li class="tool 2"><a class="tool 2"><span>积极用户</span></a></li>
            <li class="last tool 3"><a class="tool 3"><span>标签云集</span></a></li>
        </ul>
    </div>
```





```
<div id=channels-nav class=navigation-channels>
    <ul class=primary-links>
        <li class="first title 0"><span class="title 0">分类</span></li>
        <li><a style="padding-bottom: 5px">all</a></li>
        ……
    </ul>
    <span class=corner-bottom><span></span></span> </div>
```

下面是相应的一部分样式代码。

```
div.navigation-services { position: relative; width: 100%; display: block; background: #eff0f0; float: none; overflow: hidden }/*相对定位，宽为 100%，以块状的方式显示，溢出来的隐藏*/
div.navigation-services ul { padding: 0px; margin: 0px; display: inline; float: none; }
div.navigation-services ul li { padding: 0px; line-height: 1em; list-style-type: none; margin: 1px 0px 0px; display: inline; float: left; color: #414141; font-size: 91.67%; font-weight: bold; }
 /*行高为 1em，向左浮动，字体大小为 91.67 并加粗显示*/
div.navigation-services ul li a { padding: 0.5em 0.8em; margin: 0px; width: auto; display: block; background: none transparent scroll repeat 0% 0%; color: #414141; font-size: 1em; font-weight: bold; text-decoration: none; }
div.navigation-services .corner-top { position: absolute; width: 100%; background-position: -3px -3px; top: 0px; left: 0px }/*绝对定位于左上角，宽为 100%，背景图片显示的相应位置*/
div.navigation-services .corner-top span { background-position: 0px -57px }
div.navigation-channels { position: relative; width: 100%; display: block; float: none; overflow: hidden }
div.navigation-channels ul { padding: 0px; margin: 0px; width: 100%; display: inline; background: #414141; float: left; }/*内外边距都为 0 像素，以行内的方式显示，向左浮动*/
div.navigation-footer .corner-bottom { margin: 0px; background-position: -3px -12px }
div.navigation-footer .corner-bottom span { background-position: 0px -12px }
```

第 5 步，编写 B 区<div id="nav">的结构和样式代码。

下面分析网站主体部分栏目的标题区域。最外层的包含框为<div id="nav">，设置高为 40 像素，居左对齐，里面的文字大小为 14 像素。然后写一个层<div class="navigation-main" id="nav_main">，向左浮动，内容溢出来的隐藏。接着写一个无序列表<ul class="primary">，向左浮动，以行内的方式显示。上左外边距为 10 像素、15 像素。

在标题区域里面先写 3 个子标签<li>，行高为 1em，去掉其前面的小圆点样式，宽为 10.5em，以行内的方式显示，同时背景颜色为接近黑色的灰色。向左浮动，字体颜色则为白色并加粗。其中第一个为当前正在展示的，故为其设置一个类<li class="active">，设置背景颜色为蓝色。

然后后面再写 4 个<a>链接标签，放置后面的 4 个文字链接，其中第 1 个“全部”连接一张背景图等相关样式即可。整个区域分析如图 16.18 所示。



Note

掘客热点　文章队列　添加新文章　全部　新闻　视频　音频

最外面的包含框为<div id="nav">，高为40像素，居左对齐。里面写 1 个<ul class="primary">并设置向左浮动。先写3个子标签放这里的栏目切换标题。第一个背景色为蓝色，后两者为黑色。后面再写上4个<a>标签，放置后面的文字导航。

图 16.18　<div id="nav">的结构分析

结构代码如下。

```
<div style="text-align: left; height: 40px; font-size: 14px" id=nav>
    <div id=nav_main class=navigation-main>
        <ul class=primary>
            <li class=active><a class=active><span>掘客热点</span></a></li>
            <li><a class=active><span>文章队列</span></a></li>
            <li><a class=active><span>添加新文章</span></a></li>
            <a style="background-image: url(images/backew.png); width: 40px; height: 40px">全部</a>    新闻    <a>视频</a>    音频
        </ul>
    </div>
</div>
```

样式代码如下所示。

```
div.navigation-main { position: absolute; line-height: 0; width: 100%; bottom: 0px; display: inline; float: left; overflow: hidden }
div.navigation-main ul { padding: 0px; margin: 10px 0px 0px 15px; display: inline; float: left; }
div.navigation-main ul li { padding: 0px; line-height: 1em; list-style-type: none; margin: 0px 5px 0px 0px; width: 10.5em; display: inline; background: #414141; float: left; color: #fff; font-size: 91.67%; font-weight: bold; }/*行高1em，宽 10.5em，行内显示，向左浮动，字体大小 91.67%并加粗*/
div.navigation-main ul li.active { background: #247cd9; color: #ffffff }
div.navigation-main ul li a { text-align: center; padding: 0.5em; margin: 0px; display: block; background: none transparent scroll repeat 0% 0%; color: #ffffff; font-size: 1em; font-weight: bold; text-decoration: none;}
/*居左对齐，4 边的内边距为 05.em，以块状的方式显示，没有连接背景图片*/
div.navigation-main ul li.active a { background: none transparent scroll repeat 0% 0%; color: #ffffff }
```

第 6 步，分析 B 区<div class="news-summary">的框体结构，如图 16.19 所示。

从效果图中我们知道网站的主要内容都是由一个一个模块组合起来的，故只要分析其中一个模块即可。这里选择“巴黎之夜”来分析。

首先最外面的包含框为<div class="news-summary">，相对定位，底内边距为 10 像素，宽为 100%，





清除两边的浮动。里面为了得到圆角效果，先写一个<b class="roundbar">，以块状的方式显示。然后里面写了 5 个<b>标签。第 1 个<b>左右内边距为 1 像素，左右外边距为 3 像素，背景颜色为灰色，高为 1 像素，左右边框为 1 像素的灰色实线。第 2 个<b>其他不变，左右外边距设为 1 像素。第 3 个<b>在第 2 个的基础上内边距设为 0 像素。第 4 个<b>则在第 3 个的基础上外边距设为 0 像素。第 5 个<b>同第 4 个做同样的设置。这样通过层层的设置实现圆角效果。

然后写一个层<div class="roundbarfg">，背景颜色同样为灰色。里面先写一个无序列表<ul class="news-upcoming">，绝对定位，离外边框上边距离为 8 像素，左边的距离为 6 像素，居左对齐，宽为 55 像素，高为 80 像素，最后连接一张背景图。接着里面写 3 个子标签<li>，第 1 个<li class="vote-publish">设置字体大小为 24 像素，行高为 35 像素，上下两边的内边距为 15 像素和 6 像素。第 2 个和第 3 个<li class="vote" id="xvote-6">宽为 55 像素，居中对齐。

接着<div class="top">放这里的主要内容，我们将在下一步分析。最后写<b class="roundbar">，完成圆角效果，步骤刚好和前面分析的相反。

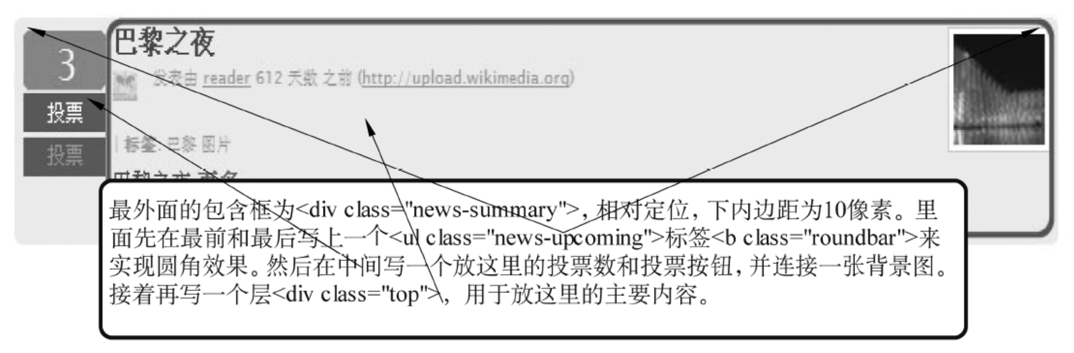


图 16.19　<div class="news-summary">的框体结构

具体结构代码如下。

```
<div id=xnews-6 class=news-summary> <b class=roundbar> <b class=roundbar1><b></b></b><b class=roundbar2>
<b></b></b> <b class=roundbar3></b><b class=roundbar4></b> <b class=roundbar5></b></b>
    <div class=roundbarfg>
        <ul class=news-upcoming>
            <li style="margin: 0px auto" class=vote-publish><a>3</a></li>
            <li id=xvote-6 class=vote><a>投票</a></li>
            <li id=xvote-6 class=vote><a style="color: #979797">投票</a></li>
        </ul>
        <div class=top></div>
    </div>
    <b class=roundbar><b class=roundbar5></b><b class=roundbar4></b> <b class=roundbar3></b><b class=
roundbar2> <b></b></b> <b class=roundbar1><b></b></b></b> </div>
```

下面是 CSS 样式代码。



Note

```
.news-summary { position: relative; width: 100%; clear: left; padding: 1px 0px 10px }
.roundbar { display: block }
.roundbar * { display: block; background: #ebebeb; height: 1px; font-size: 12px; overflow: hidden }
/*以快转的方式显示，背景颜色为灰色，高为 1 像素，溢出来隐藏*/
.roundbar1 { border-left: #f1f1f1 1px solid; padding:0px 1px;  background: #ebebeb; margin:0px 3px; border-right:
#f1f1f1 1px solid }
.roundbar2 { border-left: #f1f1f1 1px solid; padding:0px 1px; background: #ebebeb;  margin:0px 3px; border-right:
#f1f1f1 1px solid }
/*左右边框为 1 像素的灰色实线，背景颜色为灰色，左右内边距为 1 像素，左右外边距为 3 像素*/
.roundbar3 { border-left: #ebebeb 1px solid; margin:0px 1px; border-right: #ebebeb 1px solid }
.roundbar4 { border-left: #f1f1f1 1px solid; border-right: #f1f1f1 1px solid }
.roundbar5 { border-left: #ebebeb 1px solid; border-right: #ebebeb 1px solid }
.vote-publish a { padding:15px 0px 6px 0px; line-height: 35px; letter-spacing: -1px; font-size: 24px; text-decoration:
none; }/*上下内边 15 像素和 6 像素，行高 35 像素，去掉文字的下划线*/
```

第 7 步，接上一步分析<div class="top">的框体结构，如图 16.20 所示。

接着上一步分析栏目里面主要内容的结构和样式。这里最外边的包含框为<div class="top">，左外边距为 60 像素，里面主要分为 5 个层。第 1 个层为<div>，放置效果图，最右边的头像图片设置为有浮动。第 2 个层为<div class="toptitle">，放置这里文章的标题“巴黎之夜”，设置字体为 16 像素。第 3 个层为<div class="news-submitted">，底外边距为 3 像素，字体大小为 85%。里面的文字信息分别写在<a>或者<span>里面，并写上相对应的样式即可。这里就不再详细阐述。第 4 个层为<span class="news-body-text">，宽为 200 像素，上下内边距为 10 像素，左内边距为 40 像素，字体大小为 110%，里面的“更多”则加粗显示。第 5 个层为<span class="news-details">，用于放置对文章相关操作的文字或图片链接，读者可以参考下面的结构和样式代码。

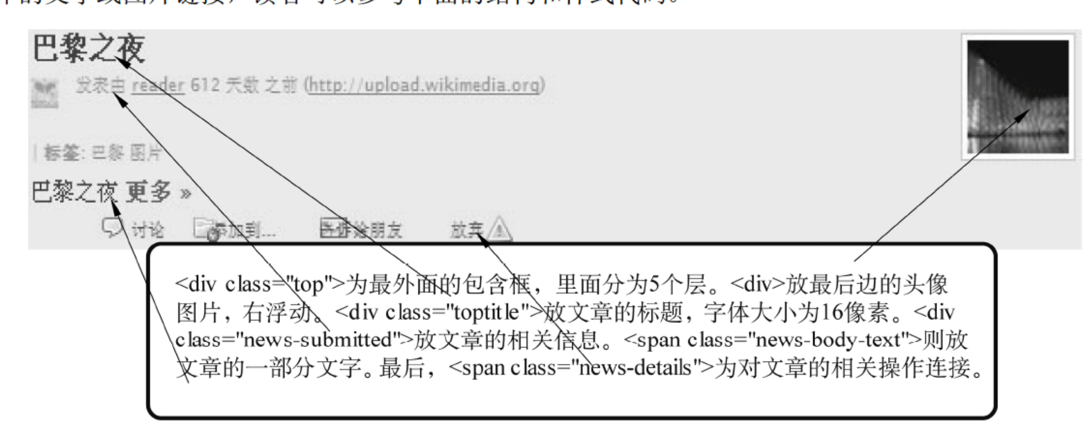


图 16.20　<div class="top">的框体结构

结构代码如下。





```
<div class=top>
    <div style="float: right; margin-right: 5px"><img src=""></div>
    <div id=ls_thetitle-6 class=toptitle><a>巴黎之夜</a> </div>
    <div class=news-submitted> <a><span id=ls_avatar-6><img src=""></span></a> <span id=ls_posted_by-6>发表由 </span> <a><span id=ls_link_submitter-6></span></a> <span id=ls_timeago-6>天数 之前 </span> <span id=ls_story_link-6></span> <span id=ls_tags-6><b><a>标签</a></b>:</span> </div>
    <span class=news-body-text><a>更多</a> <br>
    </span> <span class=news-details> <span id=ls_comments_url-6> <img src=""> <a class=comments >讨论</a>   </span> <img src=""> <a>添加到</a>  <span id=ls_recommend-6> <img src=""> <a>告诉给朋友</a>   </span> </span> </div>
```

样式代码如下。

```
.top { padding-bottom: 0px; padding-left: 5px; padding-right: 0px; margin-left: 60px; padding-top: 0px }
.news-submitted { margin-bottom: 3px; color: #999999; font-size: 85% }
.news-submitted a { color: #999; text-decoration: underline }
.news-submitted a:hover { text-decoration: none }
.news-submitted img { z-index: 1; position: relative; margin-top: 6px; float: left; margin-right: 10px }
/*堆叠顺序为 1，相对定位，上右外边距为 6 像素和 10 像素，向左浮动*/
.news-body-text { margin: 10px 0px 0px; width: 200px; padding-right: 40px; font-size: 110% }
/*上外边距为 10 像素，宽为 200 像素，右内边距为 40 像素，字体大小为 110%*/
.news-body-text a:link { color: #666666; font-weight: bold; text-decoration: none }
.news-body-text a:visited { color: #666666; font-weight: bold; text-decoration: none }
.news-details { margin: 0px; font-size: 85% }
.news-details a:link { color: #626262; text-decoration: none }
.news-details b { color: #c00 }
* html .news-details { padding-right: 136px }
```

第 8 步，编写 C 区 <div class="featurebox">的结构和样式。

下面分析的是侧栏的“分类”栏目，该栏目最外面的包含层为<div class="featurebox">，下外边距为 15 像素，上左内边距则为 15 像素和 10 像素。连接一张背景图。这里面主要分为 3 个层，第 1 个层为<div class="tlb">，上、右、下、左 4 边的内边距分别为 3 像素、10 像素、5 像素和 10 像素。而上右外边距为-15 像素和-10 像素。里面先写一个<span>标签，里面放置一个方向向下的小箭头图标图片，并设置为向右浮动。而标题则写在一个<a>里面，大小为 14 像素并加粗显示。

第 2 个层为<div id="s1">，放置这里的段落文字，设置字体大小为 12 像素，颜色为蓝色。第 3 个层为<div class="tlb2" id="maintab">，下内边距为 20 像素，里面先写两个<span>标签，放置这里的“登录”“注册”文字，设置为居中对齐，宽为 60 像素，高为 21 像素，以块状的方式显示，向左浮动，字体颜色为白色，鼠标指针以手形的方式显示，右外边距为 8 像素，最后分别连接一张颜色深浅不同的背景图。 而“更多”则写在一个<a>标签里面并设置有浮动即可，如图 16.21 所示。





Note

分类

掘客.com是一个以信息分享为主的 Web 2.0 社区网站，包括文章、视频、论坛三大分。掘客.com 的主要运行机制和功能：

1. 所有文章由用户提交，由用户票掘，票数越多的文章将显示在首页。

2. 网络书签功能。用户资料中包括了每位用户提交的所有信息，而且对应的每一项均有输出。

注册　登录　更多......

<div class="featurebox">为这里最外层的包含框，上左内边距分别为15像素和10像素。连接一张背景图。里面分为3个层：<div class="tlb">放这里的标题“分类”。<div id="s1">放这里的段落文字，字体大小为12像素，蓝色。<div class="tlb2">下内边距为20像素。里面再写两个<span>标签连接相应的背景图，“更多”放在<a>标签里面。

图 16.21　<div class="featurebox">的结构

结构代码如下。

```
<div class=featurebox>
    <div class=tlb><span><a><img src=""></a></span> <a>分类</a> </div>
    <div id=s1>
        <div class=tog></div>
    </div>
    <div id=maintab class=tlb2> <span><a style="text-decoration: none">注 册 </a></span> <span class=selected><a style="text-decoration: none">登录</a></span>     <a><b>更多……</b></a> </div>
</div>
```

具体样式代码如下。

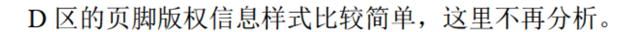
```
.featurebox { margin: 0px 0px 15px; background: url(featurebox_bg.png) no-repeat 100% 100%; color: #247cd4; padding: 15px 0 0 15px }
.featurebox p { border: medium none; margin: 0px 0px 1em; color: #444;}
.featurebox a { color: #247cd4; text-decoration: none }
.featurebox a:hover { color: #666666; text-decoration: underline }
.featurebox li a { line-height: 2em; padding-left: 20px; margin-left: -10px }
.featurebox ul { margin-bottom: 10px; margin-left: 10px }
.featurebox ul a { margin-bottom: 5px }
.tlb { margin: -15px -10px 0px; background: url(featurebox_h.png) no-repeat 0px 0px; padding: 3px 10px 5px 10px }/*上右内边距值分别为-15px 和-10px，背景图不重复，上右下左 4 边内边距值分别为 3px、10px、5px、10px，以调整内容显示位置 */
.tlb2 { padding-bottom: 20px; margin: 5px 0px 0px; }
.tlb2 span a { text-align: center; width: 60px; display: block; background: url(exp_on.png) no-repeat 0px 0px;
```







```
float: left; height: 21px; color: #ffffff; cursor: pointer; margin-right: 8px; padding-top: 2px }
    /*居中对齐，宽 60 像素，以块状方式显示，背景图片不重复，鼠标指针以手形的方式显示*/
    .tlb2 span.selected { text-align: center; color: #666; cursor: text; font-weight: bold }
    .tlb2 span.selected a { text-align: center; width: 60px; display: block; background: url(exp_down.png) no-repeat
0px 0px; float: left; height: 21px; color: #666; cursor: text; font-weight: bold; margin-right: 8px; padding-top: 2px }
```



D 区的页脚版权信息样式比较简单，这里不再分析。

软件开发视频大讲堂



循序渐进，实战讲述

375个应用实例，32小时视频讲解，基础知识→核心技术→高级应用→项目实战

海量资源，可查可练

◎ 实例资源库　◎ 模块资源库　◎ 项目资源库

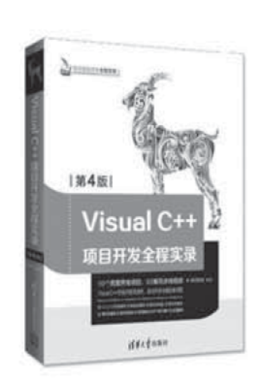
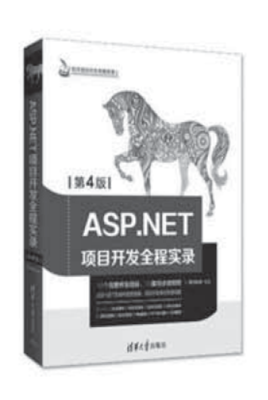
◎ 测试题库　◎ 面试资源库　◎ PPT课件

（以《Java从入门到精通（第4版）》为例）

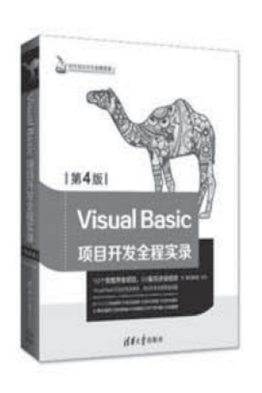
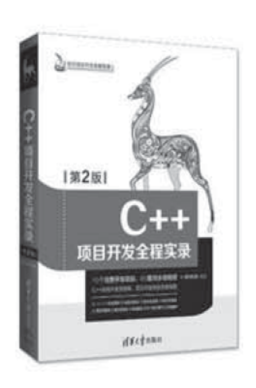

软件项目开发全程实录


第4版
Java
项目开发全程实录

第2版
C语言
项目开发全程实录

第4版
PHP
项目开发全程实录

第4版
Visual C++
项目开发全程实录

第4版
ASP.NET
项目开发全程实录

第4版
C#
项目开发全程实录

Java Web
项目开发全程实录

第4版
Visual Basic
项目开发全程实录

第2版
C++
项目开发全程实录